REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	16 Feb '96	Final Report is Feb.'94 to 15 Aug.'95

4. TITLE AND SUBTITLE
Dynamic Resource Allocation System

5. FUNDING NUMBERS
DLA 900-91-C-1482

6. AUTHOR(S)
Dr. Denis Gracanin
Dr. Stanford Smith
Dr. Padmini Srinivasan
Mr. Al Steward
Dr. Kimon Valavanis
Dr. Theodore Williams

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
A-CIM Center
University of Southwestern Louisiana
P.O. Box 44932
Lafayette, LA 70504-4932

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Logistics Agency
AQPOT Room 3135
Technical Enterprise Team
8725 John J. Kingman Road
Fort Belvoir, VA 22060-6221

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This report does not contain confidential or classified information

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A dynamic resource allocation system has been developed at the USL A-CIM Center. This technology uses colored Petri Nets coupled with Object Oriented Databases to model apparel manufacturing cells and simulate their operation.
The system is intended to recieve notification when a change in manufacturing resources or production requirements occurs. It then facilitates the determination of appropriate schedule changes to overcome or lessen the impact of the changes.

19960726 061

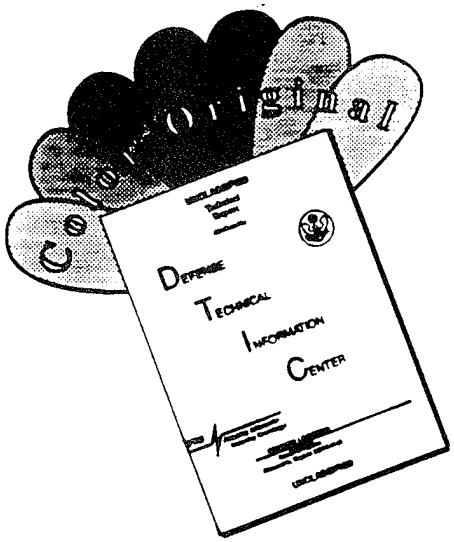
14. SUBJECT TERMS
Dynamic Resource Allocation, Dynamic Scheduler, Color Petri Net

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Dynamic Resource Allocation System
DLA900-91-C-1482 Phase III

Denis Gracanin, Stanford A. Smith, Padmini Srinivasan, Al Steward,
Kimon P. Valavanis and Theodore Williams, II

February 16, 1996

Contents

Abstract

A major problem faced by the typical apparel manufacturer in day-to-day operations is the need for an effective scheduler to determine actions required due to operator and/or machine non-performance. Current scheduling systems are static by design and provide a gross production schedule based on order demands and resources available. Thus, the current systems provide a plan for the season, month, week, and/or day, but assume that the manufacturing resources are available on either a constant or average basis. Should the resources become unavailable due to absenteeism, machine breakdown, etc., the static schedulers must be rerun to determine a new plan. In most cases, the current (dynamic) information is not available, as updates to the (system) database(s) are only done daily or even less frequently. Where dynamic databases do exist, the current scheduling systems typically provide an alternative tool which allows management to try different solutions and simulate the effectiveness of the choice. This method is limited by the ability of management to identify optimal choices, simulate effective results, and communicate the problem and chosen solution to the company's human and physical resources.

The USL A-CIM Center has solved this problem by developing an automated production planning, scheduling and reactive control system, with dynamic error detection and accommodation capabilities, called the Dynamic Resource Allocation System (DRAS); DRAS monitors resource availability in a dynamic manner and automatically identifies and implements an effective solution within the affected cell, as well as among all cells involved in the manufacturing process. Thus, given the job floor environment (machine resources, production capacities, production process description, etc.), a dynamically reconfiguarle planning and scheduling scenario is derived to accommodate machine breakdowns, production rate changes, machine reconfigurability, and in general, real-time policy changes.

The DRAS scheduler utilizes distributed databases and different classes of Petri Nets (PNs) to accomplish this because of their proven successes in modeling system concurrency, parallelism, asynchronous system operations, precedence relations, system component interactions, structural and behavioral system properties, resource sharing, conflicts and potential errors. The need to induce change is easily identified based upon the length of time the resource is not available, and the optimal change is easily identified based upon the concept of balanced work flow that most nearly matches the overall production plan determined by the static scheduler.

The DRAS system interfaces with the static scheduler being used in a generic manner and facilitates other scheduler support as well. It detects production anomalies dynamically, monitors resource availability and detects machine failure or operator unavailability by both automatic notification and operator induced notification.

The DRAS has addressed the need for flexibility in apparel manufacturing processes:

- by providing the means for flexible support of automated machines through a CIM network
- by providing the necessary communication to inform the machines that a particular bundle has arrived for processing as well as giving the details of the required processing,

- by facilitating a more flexible approach to manufacturing since the machine settings can be adjusted as frequently as necessary, and,
- by allowing the use of small bundles and supporting changes in style.

The combination of the CIM system network and the DRAS results in a powerful CIM implementation that assists in quick response to civilian market demands and aids in rapid changeover to military production by accommodating wide variation in product mix, improving operator performance, reducing management involvement in day-to-day production details and providing an enhanced information system to project Mean Time Between Failure (MTBF) and Mean Time to Recover (MTTR) statistics.

Chapter 1

Introduction

1.1 Project Rationale

Automation is the major force behind the rationalization of the production process to increase output per man-hour or output for a given set of production factors. Automation is feasible because of two innovations: the digital computer and the integrated circuit.

Two key concepts for successful automation are flexibility and integration. Flexibility allows easy and rapid reconfiguration of the production system to manufacture several different products and/or product variants, achieving a high degree of machine utilization and low degree of in-process inventory and short response time to changes in consumer preferences. Integration provides a balanced material and information flow system which results in the production of the right amount of goods at the right time for demand.

The central theme of this project is Flexible Computer Integrated Manufacturing (FCIM). The main objective is to develop a fully automated production planning, scheduling and reactive control system, with dynamic error detection and accommodation capabilities, suitable for an apparel factory/industrial plant. Given the job floor environment (machine resources, production capacities, production process description, etc.) a dynamically re-configurable planning and scheduling scenario is developed to accommodate machine breakdowns, production rate changes, reconfigurability of machines, and in general, real-time policy changes.

The major advantage of this project is threefold:

- It can be used in an apparel industrial plant.
- It may be also used for military garment manufacturing. The only changes are the inputs to the system and the initial machines/system configuration.
- It may also be adapted to non-apparel manufacturing applications.

1.1.1 The Project Environment

Modern manufacturing environments are extremely dynamic and unpredictable. The *Just-In-Time* and *Zero-Inventory* approaches to manufacturing give producers higher profits and greater leverage in the marketplace while extracting a heavy toll on the plants which actually produce the goods. Although *Computer Integrated Manufacturing (CIM)* has done much to

improve plant efficiency, plant management still bears much of the burden of running a flexible and efficient manufacturing process. This is because a flexible manufacturing system (FMS) demands from managers the ability to examine and select from an assortment of production plans which are both feasible and economical. They must also be always ready to deviate from those carefully considered plans at a moments notice in the event of equipment failure, employee absence/illness, unexpected changes in production goals, etc.. What is required of modern CIM system management, and in particular of apparel CIM system management, is the ability to formulate production plans quickly and thoroughly, working as much as 1 day in advance OR with as little as 15 minutes notice. Without assistance, this degree of preparedness and responsiveness demands large amounts of time from plant managers, and requires that they have rapid access to appropriate information.

1.1.2 The Challenges

Intelligent scheduling and reactive planning demands far more information and advice than ordinary databases and expert systems can provide. While an accurate picture of the status of each machine, operator, bundle (job), and even every scrap of material is helpful, few databases are capable of storing and making available the "knowledge" of how the production system can and should work—knowledge which is vital to maintain proper plant operation. And while expert systems provide advice and answers to management's questions, they are often far too complex to maintain accurate and useful information about the real-time status of a modern production system.

What is needed is an integration of the control and information systems for a manufacturing system, a knowledge management system which both stores and applies knowledge and information about the production system in a seamless manner that provides maximum flexibly and freedom for imaginative and innovative use of the system. The underpinning of this seamless interface is the model (or scheme) used to represent the information/knowledge about the production system. Research in knowledge representation is a field of computer science which enjoys growing support and offers tremendous potential for intelligent manufacturing systems.

Thus, pivotal tasks of this project have been the development and application of a suitable scheme for representing the status and behavior of the manufacturing cell(s) and the control and information systems to take advantage of this scheme.

1.2 The DRAS Scheduler

The DRAS scheduler is composed of two modules: The first, "Basic Function," stage develops a static Petri net model of the manufacturing cell(s) used to program a "bench version" of the DRAS Scheduler software. The scheduler selected is interfaced to initialize the DRAS Scheduler. The DRAS Scheduler provides scheduling, reactive planning, database management, and resource management functions, used to evaluate the knowledge representation scheme and refine the techniques used to model production systems.

The second, "Interface," stage accepts the bench version of the DRAS Scheduler and adds a sophisticated graphical user interface (GUI) to make its full functionality available to

the system operator. Communications and control of CIM cells (workstations) are provided.

The modeling, analysis, control, and simulation tool is Colored Petri Nets (CPN). CPNs have already been proven as an effective tool for modeling and control of automated manufacturing systems for the following reasons:

- They model concurrency, parallelism, and asynchronous system operations.
- They model precedence relations, system component interactions, structural and behavioral system properties.
- They model systems which require resource sharing.
- They model system conflicts and potential errors.
- They are a mathematical and graphical tool.

In summary, CPNs and their numerous extensions provide both a qualitative and quantitative tool for system performance evaluation, resource utilization, effect of failures on system production rate, etc.

CPNs provide automatic, standardized representation and utilization of several aspects of information required for planning and scheduling, including partial ordering of concurrent operations, temporal constraints and dependencies, resource and data requirements, and information requirements, e.g., priorities, control heuristics for constraint conflict resolution, and goals (preferences). The CPN formalism makes it possible to maintain a factory schedule integrated with an event-based control mechanism to provide a framework for reactive control. Time is implicitly represented in the precedence relationships between operations, the durations associated with setup and operations, and can also be represented absolutely as start and end times or availability times.

Colored Petri nets are amenable to specific standard analyses to determine potential problems such as starvation/deadlock situations (corresponding to the nonavailability of schedule operation), buildup of tokens (parts, material) in some segment of the system, etc. It is possible to analyze the effect of various scheduling policies and preferences:

- System Definition with respect to CPN properties,
- Avoidance of Production Halt (Liveness),
- Capacity Constraint,
- Maintenance Work-in-Process Buildup (Boundedness), and,
- Error Recoverability (Reversibility).

1.2.1 DRAS Scheduler Basic Functions

The outcome of this first phase of the DRAS Scheduler project is a proven bench version of the scheduler with integrated PN controller and Object Database (see Fig. 2). The *Control System* performs planning, scheduling, error analysis, and reactive planning, while the *Information System* maintains and makes available knowledge about the system and the status of its components.

Scheduler Functionality

Task 1: Static Petri Net Model of the System Knowledge representation for machine intelligence involves identification, representation, and utilization of knowledge in problem solving. The generation of plans takes into account the full complexity of the environment modeled. A complete and accurate model of the various relationships of interacting components, processes, and production facilities is built to simulate the behavior of the system.

To decide which activity to perform, one must know:

- the operations (transformations applied to materials),
- the precedence relations of operations, the resources of the system (materials, machines, tools, information, fixtures, etc.),
- the duration of operations/demand for resources,
- the mix and number of jobs to be scheduled,
- the constraints on the system (ordering of operations, number of resources, capacity constraints, constraints on usage of resources, etc.),
- the substitutability of resources, and,
- the scheduling objectives or policies (minimum idle time, minimum work-in-process, etc.).

Task 2: Resource/Jobs/Process Scheduler A static scheduler takes as input a detailed system description, expected job mix, operator characteristics, and intelligence gleaned from shop floor experts. It provides as output a pre-schedule (operation sequence), which is used as the initial expected behavior of the DRAS Scheduler. A general format is specified for the interface between static schedulers and the DRAS Scheduler.

Task 3: Database Management An object database serves as the state representation enabling transitions to occur in the scheduler. Information management technology is applied to:

- the control information or meta-data (system and control definitions),
- the resource database (managing cell units, operators, etc.), and,
- inventory (input materials, work-in-process, etc.).

Designing the database is an implementation project, while the knowledge representation for decision support requires more theoretical study.

Task 4: Resource Management There are three major system resources:

- input material,
- operator availability, and,
- production machines.

The resource manager must constantly monitor the availability of input material and status of machines on a real-time basis. This is essentially a specialization/optimization of the scheduler's more general functionality to provide specific operations as efficiently as possible.

Task 5: Error Analysis and Reactive Planning An intelligent system needs the ability to react to changing circumstances. Errors may be introduced during simulation to develop and evaluate recovery strategies. Errors may also occur online, during plan execution, in response to real problems. Inputs to an error handler include definitions of error conditions, descriptions of expectations regarding error (MTBF, MTTR), descriptions of system actions for error handling and recovery policies. Outputs include evaluations of various error recovery policies based on simulation. The system is used to directly coordinate error recovery on the shop floor based on successful simulation. The scheduler can (in an operational environment) control recovery actions directly or suggest recovery actions to the operator on the shop floor.

DRAS Scheduler Interface

The outcome of this second phase is a tested, *production version* of the scheduler with full communications to the system operator, Leadtec system, and the CIM cells. The *System Operator Interface* is supported by a low-level Graphical User Interface (GUI) built to access and manipulate the CPN controller and object database.

Task 6: System Monitor To make the scheduler functionality accessible to management and operators, a sophisticated user interface is provided having an iconic view of the production system. A low-level, programmable, Graphical User Interface (GUI) is constructed and linked to the scheduler. This GUI may then be programmed to provide whatever interfaces are desired to monitor and manipulate the scheduler.

The System Monitor is a simple, first interface that is both useful and easy to program into the GUI. It shows the overall status of the system as a collection of labeled icons and allows the operator to "click" on a particular icon to get more information about some aspect of the system. It offers a visual explanation of the system and allows the operator to monitor system conditions easily and intuitively. In addition to a *static picture* of the system status, the system monitor also shows the flow of information, parts, and resources through the system as a schedule is executed. This flow (real-time or simulated), is the *dynamic picture* of the system which conveys a true picture of what the system is actually doing at any moment.

Task 7: System Operator Interface As an extension to the System Monitor, the System Operator Interface makes the full functionality of the scheduler available to the operator via interfaces built upon the programmable GUI. Operations that are supported include defining a new CIM cell (workstation), identifying a CIM cell (function, capacity, location, etc.), define a new job (product code, size, quantity, priority, etc.), asserting control of a cell unit (activate a cell, deactivate a cell, re-route a process, change a job priority, etc.), generate reports (for supervisors and managers).

Task 8: Cell Controller Protocol The Cell Controller Protocol facilitates the exchange of control signals and messages among the CIM system, operator, and both master and cell controller. Control signals for CIM cells can initiate automatic cell actions based on plans, schedules, and system state, while messages to human operators may warn of impending problems and/or request intervention.

Task 9: Interface Protocol to Leadtec System A second protocol may allow the DRAS Scheduler to communicate with the Leadtec system at a selected plant. This includes an interface to the Leadtec software running on a "host" computer, as well as the Leadtec barcode reader used to track bundles through the system.

Chapter 2

Basic Concepts

2.1 Introduction

2.2 Petri Nets

Theoretical methodologies and applications of PNs have been developed for modeling, analysis and performance evaluation of production control systems, factory automation and discrete event dynamic systems [7, 10, 28, 39, 48, 49, 51]. Detailed studies related to the PN history, application areas and control of automated manufacturing systems may be found in [3, 17, 43].

Directions in which ordinary PNs are modified include:

- Imposing additional constraints on the ordinary PN model, thus making the analysis easier, and,
- Adding enhancements or new semantics to the ordinary PN model, thus increasing the expressive power.

Enhancements and modifications of ordinary PNs have been introduced to increase their modeling capabilities. They include the theory of modified PNs [7], Prot nets [10], timed decision free PNs [19], many–sorted High–level nets [8], rainbow nets [34], stochastic nets [2, 27], colored adaptive structured PNs [23], Predicate/Transition nets (PrT-nets) [44], Colored PNs (CP-nets) [33], as well as many additional general and/or specific (place, transition) modifications [4, 3, 13, 15, 36, 39, 40, 45, 55, 56, 61, 66, 67]. It is important to emphasize that most of the net models were basically designed with a single or narrow application area in mind. A breakthrough occurred when Predicate/Transition nets were introduced to incorporate in a formal manner the concept of "individuals with changing parameters and relations" into net theory [21, 22].

CP–nets were introduced to overcome limitations related to the calculation of linear place–invariants (S–invariants) in PrT–nets (which contain "free variables" over sets of colors) [31, 32]. CP–nets attach explicitly a set of possible token–colors to each place and a set of possible occurrence–colors to each transition. Quoting [31], CP-nets differ from PrT–nets in the following ways:

- The set of possible token–colors at a place is explicitly defined,
- The number of tokens added or removed at a given place may be different for two occurrence–colors of the same transition, and,
- The set of allowable expressions and predicates is not explicitly defined but if desired this can be done by means of a many–sorted algebra, from which the allowable expressions, predicates, functions and sets can be built up.

Moreover, it has also been shown in [31], how to combine the qualities of PrT–nets and CP–nets into a single net model, called High–level PNs. (The name created confusion since this term was also used as a generic name for PrT–nets, CP–nets, Relation nets, etc.).

Recently, PNs and their modifications have also been used as a tool for performance analysis and evaluation of decision making organizations associated with Command, Control, and Communications (C^3) systems [54, 60, 64], Artificial Intelligence (AI) planning and heuristic problems [46, 47], and coordination of intelligent machines [62, 63].

Important properties that have contributed to the value of PNs as a modeling tool, include: model clarity and ease of representation, ability to model nondeterminism, conflicts, timing information, resource sharing, concurrency, parallelism and control of asynchronous operations [41, 43, 57, 68]. The existence of qualitative analysis and visualization techniques, the ability to link directly the net structure properties (liveness, consistency, boundedness, connectivity) to desirable real–time/real–life system performance criteria, and the ability to utilize the structure of the net model (partial ordering) to characterize system causality, are additional properties which justify further the wide utilization of PNs as a modeling tool. The very extensive list of references in [43] provides further justification for the applicability of PNs to various diverse applications.

System design with PNs currently follows three principles [53]:

- Distinguish system components and designate every component to be either passive or active. The passive components store things or make them visible. The active components refer to produce, transport or change things,
- Systematic movement from PNs consisting of active and passive components to PNs that model dynamic behavior, and,
- During the development of PN representation, there are two ways of developing a model:
  - Replacing a net element with a corresponding subnet, and,
  - Adding a net element.

There are certain limitations related to the modeling power of existing classes of PNs. These limitations become more obvious when one deals with complex, dynamic, hierarchically intelligent systems:

- System descriptions (based on existing classes of PN) result in huge models with a corresponding drastic reduction in their manageability,

- It becomes extremely difficult to verify the system model structural properties,
- For hierarchical systems, pertinent composition/decomposition rules for the PN models are either not well defined, or are not effective in simplifying the PN operation, and,
- While PNs clearly provide a useful operational model, their applicability is limited for system specification purposes mainly due to lack of abstraction concepts.

To be more specific, current hierarchical approaches related to PN modeling utilize the concept of "subnets". This results in higher level nets with a smaller number of places. Analysis at the subnet level is propagated to the higher level net. However, the firing/operation of the total net is not simplified by the identification of the node clusters which constitute the subnets.

2.3 PN Definitions

An ordinary PN consists of a fixed number of places and transitions with tokens distributed over places. Each transition has an associated set of input places and a set of output places. When every input place of a transition has enough tokens, the transition is enabled and may fire. When a transition is fired, a token is consumed (deleted) from every input place and produced (added) to every output place.

The structure of an ordinary PN as well as various definitions are taken from [49] in order to provide a foundation for the similar definitions for the PPN.

Definition 2.1 (Petri Net) *A Petri net structure, C, is a four–tuple:*

$$C = (P, T, I, O)$$

where:

P : *A finite set of places* $P = \{p_1, \ldots, p_n\}$, $n \geq 0$,

T : *A finite set of transitions* $T = \{t_1, \ldots, t_m\}$, $m \geq 0$,

I : *An input function, a mapping from transitions to bags (multisets) of places,* $I : T \rightarrow P^{\infty}$,

O : *An output function, a mapping from transitions to bags of places,* $O : T \rightarrow P^{\infty}$.

The set of places and the set of transitions are disjoint:

$$P \cap T = \emptyset$$

For a given PN, a corresponding graphical representation, the PN graph is:

Definition 2.2 (Petri Net Graph) *A Petri net graph G is a bipartite directed multigraph:*

$$G = (V, A)$$

where:

V : *A finite set of vertices* $V = \{v_1, \ldots, v_s\}$,

A : *A finite bag of directed arcs* $A = \{a_1, \ldots, a_r\}$, $a_i = (v_j, v_k)$, $v_j, v_k \in V$.

The set V *is partitioned in two disjoint sets* P *and* T *such that:*

$$V = P \cup T, \quad P \cap T = \emptyset$$

and for each directed arc $a_i = (v_j, v_k)$, *either:*

$$v_j \in P \quad \wedge \quad v_k \in T$$

or:

$$v_j \in T \quad \wedge v_k \quad \in P$$

For a place $p_j \in P$ and a transition $t_k \in T$ the folowing holds:

- t_k is the input transition of p_j if there exists a directed arc $a_i \in A$ from t_k to p_j,
- t_k is the output transition of p_j if there exists a directed arc $a_i \in A$ from p_j to t_k,
- p_j is the input place of t_k if there exists a directed arc $a_i \in A$ from p_j to t_k, and,
- p_j is the output place of t_k if there exists a directed arc $a_i \in A$ from t_k to p_j.

Definition 2.3 (Marking) *A marking* μ *of a Petri net* $C = (P, T, I, O)$ *is a function from the set of places* P *to the nonnegative integers* $\mathcal{N}$*:*

$$\mu : P \rightarrow \mathcal{N}$$

Each place may contain tokens. A token is a primitive concept like place or transition and it is used to define the execution of a PN. A marking indicates the number of tokens in each place.

Definition 2.4 (Marked Petri Net) *A marked Petri net:*

$$M = (C, \mu) = (P, T, I, O, \mu)$$

is a Petri net structure C *and a marking* μ.

A PN executes by firing transitions. A transition fires by removing tokens from its input places and creating new tokens which are distributed to its output places (one token for each arc).

Definition 2.5 (Enabled Transition) *A transition* $t_j \in T$ *in a marked Petri net* $C = (P, T, I, O)$ *with marking* μ *is enabled if for all* $p_i \in P$*:*

$$\mu(p_i) \geq \#(p_i, I(t_j))$$

Firing an enabled transition t_j *results in a new marking* μ' *defined by:*

$$\mu'(p_i) = \mu(p_i) \quad - \quad \#(p_i, I(t_j)) \quad + \quad \#(p_i, O(t_j))$$

where $\#(p_i, I(t_j))$ *denotes the multiplicity of a place* p_i *in the bag of input places of a transition* t_j, *and* $\#(p_i, O(t_j))$ *denotes the multiplicity of a place* p_i *in the bag of output places of a transition* t_j.

Firing of transitions changes the marking of a PN.

Definition 2.6 (Immediately Reachable Marking) *For a Petri net $C = (P, T, I, O)$ with marking μ, a marking μ' is immediately reachable from μ if there exists a transition $t_j \in T$ such that:*

$$\delta(\mu, t_j) = \mu'$$

where δ is a next–state function which gives the marking resulting from firing a transition t_j under marking μ.

Definition 2.7 (Reachability set) *The reachability set $R(C, \mu)$ for a Petri net $C = (P, T, I, O)$ with marking μ is the smallest set of markings defined by:*

1. $\mu \in R(C, \mu)$.
2. *If $\mu' \in R(C, \mu)$ and $\mu'' = \delta(\mu', t_j)$ for some $t_j \in T$, then $\mu'' \in R(C, \mu)$.*

Definition 2.8 (Safeness) *A place $p_i \in P$ of a Petri net $C = (P, T, I, O)$ with initial marking μ is safe if for all $\mu' \in R(C, \mu)$:*

$$\mu'(p_i) \leq 1$$

A Petri net is safe if each place in that net is safe.

Definition 2.9 (Boundedness) *A place $p_i \in P$ of a Petri net $C = (P, T, I, O)$ with initial marking μ is k–safe if for all $\mu' \in R(C, \mu)$:*

$$\mu'(p_i) \leq k$$

A Petri net is k–safe if each place in that net is k–safe.

Note that safeness is a special case of boundedness where $k = 1$.

Definition 2.10 (Strictly Conservative Petri Net) *A Petri net $C = (P, T, I, O)$ with initial marking μ is strictly conservative if for all $\mu' \in R(C, \mu)$:*

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i)$$

Definition 2.11 (Conservative Petri Net) *A Petri net $C = (P, T, I, O)$ with initial marking μ is conservative with respect to a weighting vector $\vec{w}$, $\vec{w} = (w_1, \ldots, w_n)$, $n = |P|$, $w_i \geq 0$, if for all $\mu' \in R(C, \mu)$:*

$$\sum_{i=1}^{n} w_i \cdot \mu'(p_i) = \sum_{i=1}^{n} w_i \cdot \mu(p_i)$$

Definition 2.12 (Liveness) *A Petri net $C = (P, T, I, O)$ with initial marking μ has these levels of liveness:*

Level 0 : A transition t_j is live at level 0 if it can never be fired.

Level 1 : A transition t_j is live at level 1 if it is potentially fireable; that is, if there exists a $\mu' \in R(C, \mu)$ such that t_j is enabled in μ'.

Level 2 : A transition t_j is live at level 2 if for every integer n there exists a firing sequence in which t_j occurs at least n times.

Level 3 : A transition t_j is live at level 3 if there is an infinite firing sequence in which t_j occurs infinitely often.

Level 4 : A transition t_j is live at level 4 if for each $\mu' \in R(C, \mu)$ there exists a firing sequence σ such that t_j is enabled in $\delta(\mu', \sigma)$, where $\delta(\mu, \sigma)$ is an "extended" next-state function which gives the marking resulting from firing a sequence of transitions, $\sigma = t_{i_1} \cdots t_{i_s}$, one after another, under marking μ.

A transition which is live at level 0 is dead. A transition which is live at level 4 is live. A PN is live at level i if every transition is live at level i.

Definition 2.13 (The Reachability Problem) *Given a Petri net C with marking μ and a marking μ', is $\mu' \in R(C, \mu)$?*

Definition 2.14 (The Coverability Problem) *Given a Petri net C with initial marking μ and a marking μ', is there a reachable marking $\mu'' \in R(C, \mu)$ such that $\mu'' \geq \mu'$?*

This concludes the description of the ordinary PNs and their properties.

2.4 Modified PNs

Due to some structural characteristics, the modified classes of PNs have some characteristics which are not present in ordinary PNs generally [16], due to the additional constraints imposed on the ordinary PN model.

Definition 2.15 (State graph) *A Petri net $C = (P, T, I, O)$ is a state graph if and only if every transition has exactly one input and one output place.*

Definition 2.16 (Event Graph) *A Petri net $C = (P, T, I, O)$ is an event graph if and only if every place has exactly one input and one output transition. It is also called transition graph or marked graph.*

Definition 2.17 (Conflict–free Petri Net) *A Petri net $C = (P, T, I, O)$ is conflict-free if and only if every place has at most one output transition.*

If there is a place with more than one output transition, then a conflict exists and it is denoted as a pair of place p, and a set of output transitions of the place p, $\{t_{i_1}, t_{i_2}, \ldots\}$ as:

$$(p, \{t_{i_1}, t_{i_2}, \ldots\})$$

Definition 2.18 (Free–choice Petri Net) *A Petri net $C = (P, T, I, O)$ is a free-choice if and only if for every conflict $(p_i, \{t_{i1}, t_{i2}, \ldots\})$ none of the transitions in the conflict possess an input place other than p_i.*

Definition 2.19 (Extended Free–choice Petri Net) *A Petri net $C = (P, T, I, O)$ is an extended free-choice if and only if for every conflict $(p_i, \{t_{i1}, t_{i2}, \ldots\})$ all transitions in the conflict have the same set of input places.*

Definition 2.20 (Simple Petri Net) *A Petri net $C = (P, T, I, O)$ is a simple Petri net if and only if each transition is affected by at most one conflict.*

Definition 2.21 (Pure Petri Net) *A Petri net $C = (P, T, I, O)$ is a pure Petri net if it has no self–loop, i.e., a place p_i and a transition t_j such that p_i is both an input and an output place to t_j.*

A place p_i is a pure place if there is no transition for which p_i is both an input and an output place.

A transition t_j is a pure transition if there is no place for which t_j is both an input and an output transition.

2.5 PN Analysis

There are several approaches used to analyze the properties of PNs:

- The reachability tree,
- Linear algebra, and,
- Reduction methods.

There are also numerous other results which are very specialized.

2.5.1 Reachability Tree

The reachability tree represents the reachability set of a PN. Each node corresponds to one of the reachable markings, the root node being the initial marking. Each (directed) arc represents the firing of a transition which is enabled by the marking represented by the originating node, and a destination node represents the marking after the firing of the transition. Every path in the reachability tree represents a sequence of possible transition firings. For a PN with an infinite reachability set, the reachability tree is also infinite. Therefore, in order to make this usable, a modification, called the coverability tree, is introduced. There is a special symbol, ω, which can be thought of as "infinity" and which represents a number of tokens which can be made arbitrarily large. The finite coverability tree can be constructed by using w and the reachability tree.

There are several results which follow immediately from the coverability tree.

- A PN is bounded if and only if the symbol ω never appears in its reachability tree, i.e., the PN is a finite state system. For bounded PNs, the reachability tree and coverability tree are the same.
- If the weighted sum is the same for all nodes in the coverability tree, the PN is conservative. Note that the weight corresponding to ω must be 0.

- The coverability problem can be solved by checking if there exists a node such that corresponding marking is covered.

The reachability (coverability) tree cannot be used, in general, to solve the reachability or liveness problems, because the symbol ω represents a loss of information. Also, it can be very large even for small PNs.

2.5.2 Linear Algebra

This approach is based on a matrix view of PNs. Input and output functions are represented by two matrices, $\vec{D}^-$ and $\vec{D}^+$. Each matrix has m rows, one for each transition, and n columns, one for each place. It is defined as:

$$\vec{D}^-[j,i] = \#(p_i, I(t_j))$$

$$\vec{D}^+[j,i] = \#(p_i, O(t_j))$$

Also, the marking μ can be represented as a vector $\vec{\mu}$, where:

$$\mu_i = \vec{\mu}[i] = \mu(p_i)$$

The transition t_j is represented by the unit m–vector $\vec{e}_j$ and it is enabled if:

$$\vec{\mu} \geq \vec{e}_j \cdot \vec{D}^-$$

Consequently:

$$\delta(\mu, t_j) = \vec{\mu} + \vec{e}_j \cdot \vec{D}$$

where:

$$\vec{D} = \vec{D}^+ - \vec{D}^-$$

For a given weighting vector $\vec{w}$:

$$\vec{D} \cdot \vec{w} = \vec{0}$$

Consequently, a PN is conservative if and only if there exists a positive vector $\vec{w}$ such that $\vec{D} \cdot \vec{w} = \vec{0}$. This can also be used to find $\vec{w}$. A reachability problem reduces to solving an equation:

$$\vec{\mu}' = \vec{\mu} + \vec{x} \cdot \vec{D}$$

Unfortunately, the matrix approach has several problems:

- The matrix $\vec{D}$ itself does not properly reflect the structure of PN, e.g, transitions which have the same input and output place are represented in the same position of the D^- and D^+ matrices and have 0 in this position in the $D = D^+ - D^-$.
- The solution for the reachability problem is necessary, but not sufficient, and,
- There is a lack of sequencing information.

2.5.3 Reduction Methods

This approach is used to replace big PNs with smaller and simpler ones. These simplified nets are not equivalent to the original ones but have some properties preserved. Two main ways are [16]:

- Preservation of liveness and boundedness, and,
- Preservation of invariants.

Some types of reduction which preserve liveness and boundedness are:

Substitution of a place : A place p_i can be substituted if it satisfies the following:

- The output transitions of p_i have no other input places than p_i,
- Place p_i is pure, and,
- At least one output transition of p_i is not a sink transition.

Implicit place : A place p_i is implicit if it satisfies the following:

- The marking of this place never forms an obstacle to the firing of its output transition, and,
- Its marking can be deduced from the marking of the other places.

Neutral transition : A transition t_j is neutral if and only if the bag of its input places is identical to the bag of its output places.

Identical transitions Two transition are identical if they have the same bag of input and the bag of output places. One of them and its corresponding arcs can be suppressed.

Preservation of invariants can be done by the following reductions:

- If t_j is a self–loop transition with respect to place p_i, then the reduction consists of:
  - Suppressing arcs from p_i to t_j and from t_j to p_i, and,
  - Suppressing transition t_j if it is isolated.
- If t_j is a pure transition with at least one input and one output place, then the reduction consists of:
  - Transition t_j is suppressed.
  - A place $p_i + p_k$ is associated with every pair of places such that p_i is the input place of t_j and p_k is the output place of t_j. The number of tokens is equal to the sum of number of tokens in p_i and p_k.
  - The input (output) transitions of $p_i + p_k$ are input (output) transitions of p_i and p_k except for t_j.

2.6 Database

Database and database technology have had a major impact in almost all areas where computers are used, including business and engineering. In general a database is a collection of related data which are known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of machine operators.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the miniworld or the Universe of Discourse (UoD). Changes to the miniworld are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be reffered as database.
- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived application in which these users are interested.

2.6.1 Database Management System (DBMS)

Object-Oriented Database

Databases and database technology have had a major impact in almost all areas where computers are used, including business and engineering. In traditional database systems there is an incompatibility between database structures and the programming language's structures. An object-oriented database systems provides persistent storage for program objects and data structures. Therefore, a complex object in a programming language can be stored permanently in an object-oriented database, thus surviving the termination of program execution; further, it can be later directly retrieved. Object-oriented database systems typically offer data structure compatibility with one or more object-oriented programming languages. The Object Database Standard ODMG-93 [11] covers all the necessary functionality required for an application to create, modify and share objects.

Chapter 3

Design and Implementation

A major problem faced by the typical apparel manufacturer in day-to-day operations is the need for an effective scheduler to determine actions required due to operator and/or machine non-performance. The Dynamic Resource Allocation System (DRAS) scheduler described in this paper is the natural outgrowth of previously funded research related to the design, development, technology transfer and installment of a CIM system in an apparel plant. The already developed CIM system provides a way for the DRAS scheduler to dynamically change a production process. The need to induce change is identified based on the length of time the resource is not available, and the optimal change is identified based upon the concept of balanced work flow that most nearly matches the overall daily production plan. The DRAS scheduler detects production anomalies dynamically, monitors resource availability and detects machine failure or operator unavailability by both automatic and operator induced notification. The daily production plan is used as a guideline to determine how to react and correct production problems. The underlying system model is based on Petri nets and object-oriented databases. Petri nets provide a dynamic model of the CIM system while object-oriented databases provide necessary information about system components and overall system activity.

3.1 Introduction

The main objective of this research has been to develop a fully automated production planning, scheduling and reactive control system, with dynamic error detection and accommodation capabilities, suitable for an apparel factory/industrial plant. Given the job floor environment (machine resources, production capacities, production process description, etc.) a dynamically re-configurable planning and scheduling scenario is developed to accommodate machine breakdowns, production rate changes, reconfigurability of machines, and in general, real-time policy changes. Such an objective is extremely complex. However, for a limited problem domain and well defined area as is an apparel plant, the design and implementation becomes simpler.

Previous work related to transfer of CIM technology to an apparel manufacturing plant [58] has been used as the basis for the development of a Dynamic Resource Allocation System (DRAS), described in this paper. A CIM system has been implemented in the Levi Strauss &

Co. plant in Wichita Falls, Texas (Figure 3.1). A LeadTec Systems, Inc. automated payroll

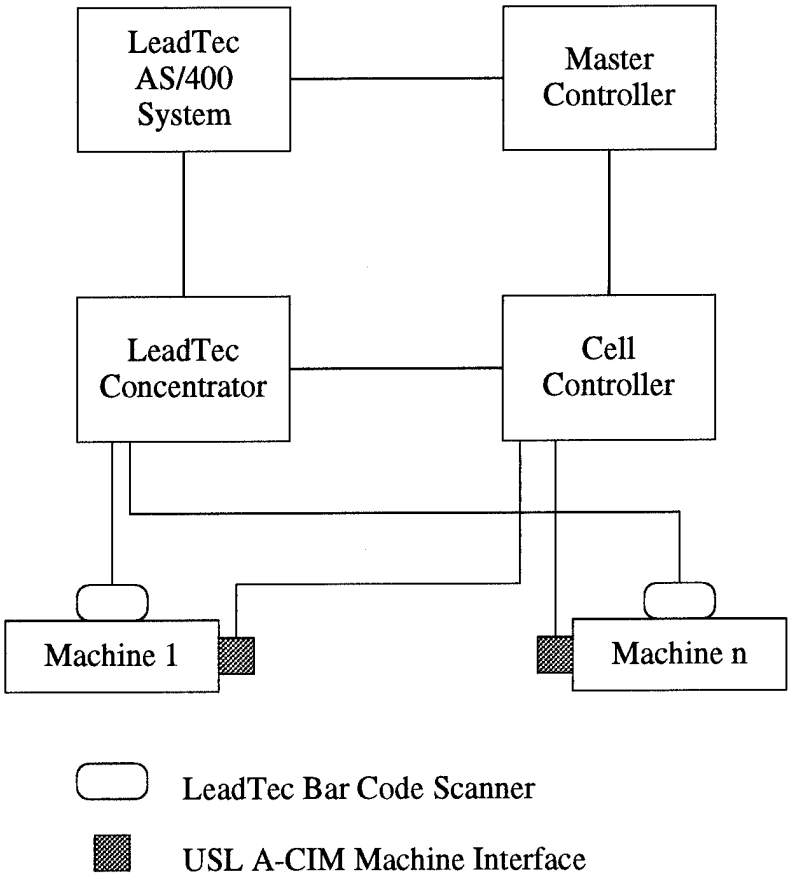


Figure 3.1: CIM System

system generates an electronic notification of bundle arrival at machines in an automated cell. This notification is used by the CIM system to retrieve the needed garment description and the appropriate sundry information which is, then, used to determine whether to instruct the machine to modify its settings or activate a different sewing program. In addition, messages are transmitted to the machine operators.

Such a system can be interfaced with a scheduler (Figure 3.2) that utilizes both human resources activity and machine activity and configuration to provide for:

Database Management : Interaction with the CIM system (payroll system and human resources activity) and management of the system data which, among others, include production history, description of the CIM system, user/machine performance, etc.

Static Petri Net Model : A Petri net Model of the CIM system is used for error analysis and reactive planning and to produce the schedule.

Resource Management : Interaction with the CIM system (machine status, part processing) and changing the machine setup when required.

Resource/Jobs/Process Scheduling : Update of the production schedule based on the current state of the system and the user input.

System Monitor : Interface to the user and data gathering from other modules to display the current status of the system and generate requested reports.

Error Analysis and Reactive Planning : "What If" system design and planning that is based on error analysis using the static PN system model. A system action simulator [20, 42] is included.

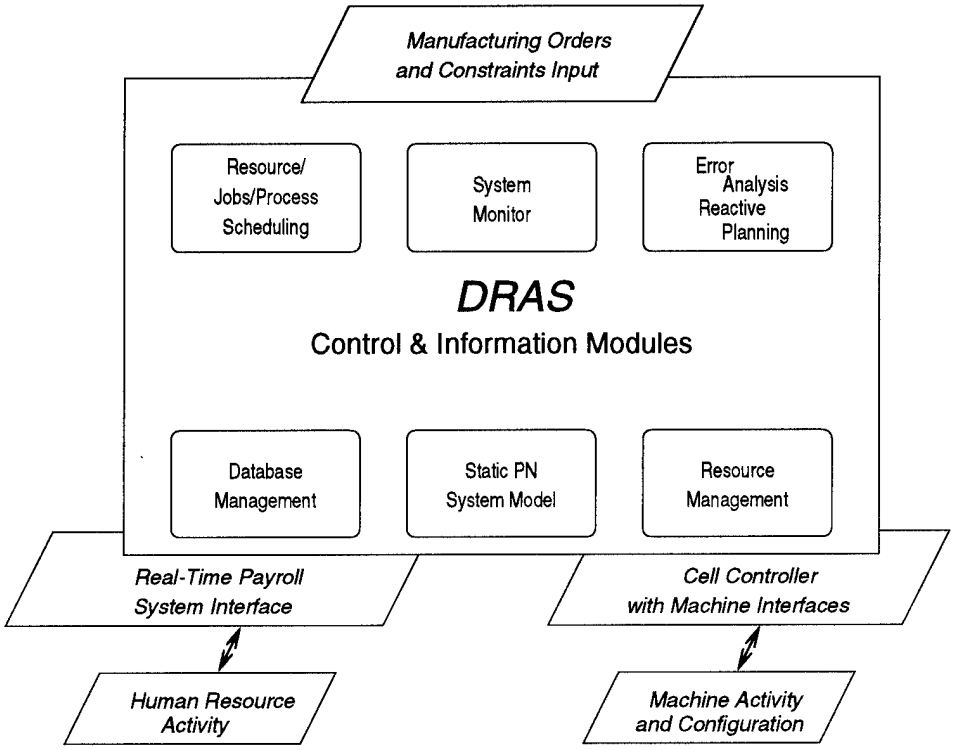


Figure 3.2: The basic structure of the DRAS system

In addition, manufacturing orders and constraints (daily production plan) are goals that need be achieved during the daily production cycle.

3.2 The DRAS Model

A manufacturing system is modeled based on the features that are important for the DRAS scheduler. The following considerations are then taken into account: manufacturing plant considerations, machine considerations, and personnel considerations.

Manufacturing plant considerations describe the overall functionality of the plant. First, a portion of the plant that is to be included in the DRAS scheduler is defined by specifying the number of separate cells and number and type of machines in each cell. For the selected

part of the plant, the level of automation is determined by specifying type of management and supervision system, inventory and storage system, transportation system, number of automated and manually operated machines, and availability of a CIM network. Then, dependence relationships between machines in each cell and between cells, are specified, as well as dependence upon sundries, supplies, human operators, etc. Finally, events in the plant are specified at the system, cell, and machine level. That includes availability of work, sundries, supplies, and task completion. In addition, system changes in machine availability, human resources, work availability and in production goals are considered.

Machine considerations are defined separately for automated and for manual machines. Both types of machines are specified in terms of capabilities, reconfigurability, current configuration and typical faults. Typical faults include level of performance loss, mean time to repair and acceptable frequency of occurrence. In addition, for automated machines electronic and manual reconfigurability is considered.

Personnel considerations are based on the performance efficiency for each task, both normal and current. In addition, availability (active and inactive) and unavailability (reason and estimated duration) are considered.

The basic structure of the model used by the DRAS scheduler is a Hierarchical Colored Petri net simulator, integrated with an object database, which stores knowledge about the status and operation of the system and provides scheduling and reactive planning services. The necessary data include description of the factory floor (floor diagram), description of the product process, available parts, and target production. Each machine is represented by the corresponding Colored Petri net. A Petri net for the whole system is generated by adding additional places and transitions to represent flow of parts among machines. The Petri net models the dynamic behavior of the system by firing transitions.

3.2.1 Petri Nets

The DRAS scheduler utilizes different classes of Petri nets (PNs) because of their proven success in modeling system concurrency, parallelism, asynchronous system operations, precedence relations, system component interactions, structural and behavioral system properties, resource sharing, conflicts and potential errors [16, 43]. In this paper, it is assumed that the reader is somehow familiar with basic PN modeling techniques. For fundamentals of PN theory, the reader is referred to [49, 52].

Important properties that have contributed to the value of PNs as a modeling tool, include: model clarity and ease of representation, ability to model nondeterminism, conflicts, timing information, resource sharing, concurrency, parallelism and control of asynchronous operations [16, 18, 69]. The PN formalism makes it possible to maintain a factory schedule integrated with an event-based control mechanism to provide a framework for reactive control. PNs are amenable to specific standard analysis to determine potential problems such as starvation/deadlock situations, buildup of tokens (parts, work-in-process) in some segments of the system, possibility of production halt, constraint maintenance, error recoverability, etc. The existence of qualitative analysis and visualization techniques, the ability to model real life system performance criteria and evaluate them based on the corresponding PN properties and the ability to characterize system causality are additional reasons for widespread use of PNs as a modeling tool. The significance of the dynamic aspects of

scheduling made possible by the Petri net model representation lies in the fact that effective computer integrated manufacturing requires the ability to efficiently schedule operations in an event-based system, including expected and unexpected occurrences.

Figure 3.3 shows how a real system may be modeled in terms of Petri nets. The system

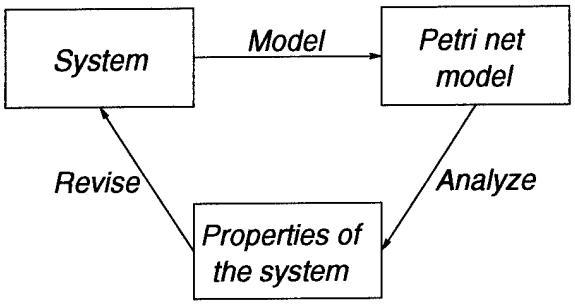


Figure 3.3: Modeling cycle

should satisfy a set of requirements. A Petri net model of the system is first constructed and analyzed. The properties of PNs like safeness, boundedness, conservation, liveness, reachability and coverability are then used to evaluate the corresponding system through the use of analysis techniques which include the reachability tree and matrix equations. The results of the analysis are then used to revise the system, if necessary, according to the requirements. The sequence: model, analyze and revise, may be repeated until the system satisfies all set requirements.

As an example, a simple model of a machine shop that can process up to two orders in parallel is considered. The corresponding PN model is shown in Figure 3.4, and is composed of the following places and transitions:

P_0 : Outside world (the rest of the PN model).

P_1 : An order is waiting.

P_2 : The order is being processed.

P_4 : The order is complete.

P_4 : The machine shop is waiting for an order.

and 4 transitions:

t_1 : An order arrives.

t_2 : Processing starts.

t_3 : Processing is complete.

t_4 : The order is sent for delivery.

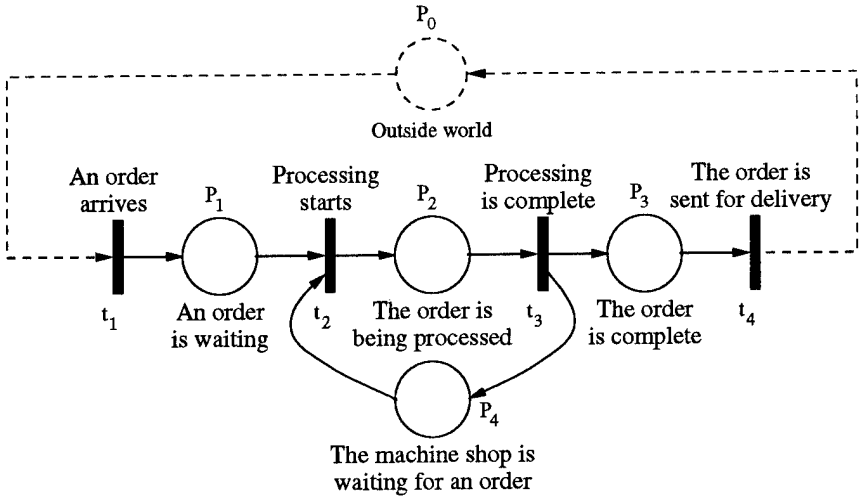


Figure 3.4: Petri Net Model

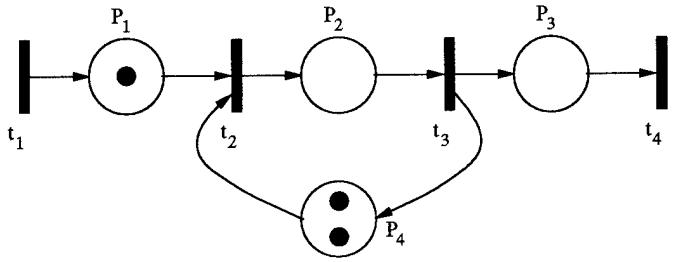


Figure 3.5: Marked Petri Net I

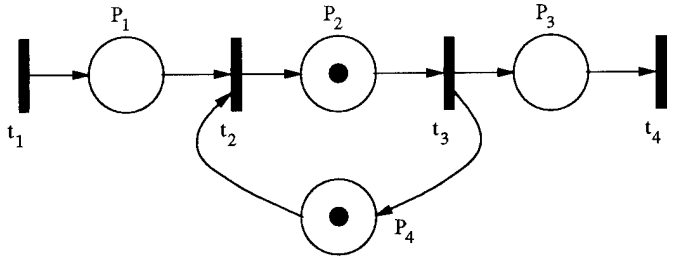


Figure 3.6: Marked Petri Net II

Figure 3.5 shows a marked Petri net when an order is waiting to be processed. Two tokens in P_4 denote that up to two orders may be processed in parallel. When a transition t_1 that correspond to the start of processing fires, a token is put in place P_2 representing that the order is being processed. Figure 3.6 shows an order being processed.

In Colored Petri nets, shown in Figure 3.7 (an enhanced class of PNs derived from ordinary PNs [31, 32]), tokens are individual, distinguishable objects, with some properties (colors) Transitions are not activated only by the presence of tokens but by the presence of

tokens of required colors.

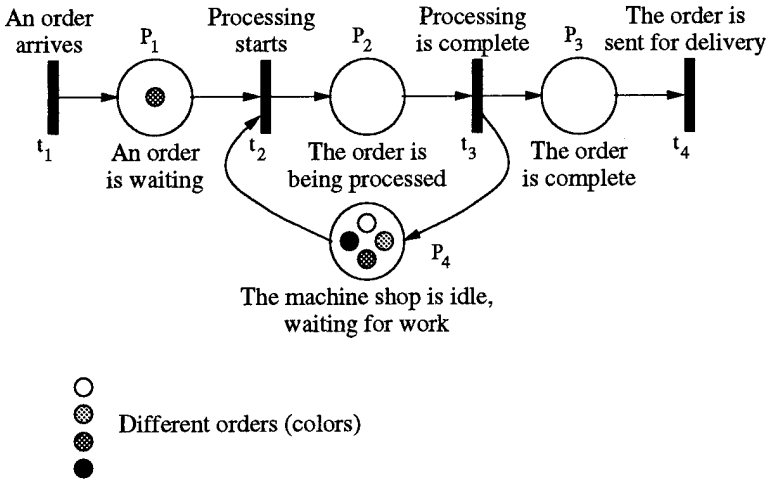


Figure 3.7: Colored Petri Net Model

Current hierarchical approaches related to PN modeling utilize the concept of "subnets" [33, 29]. This results in higher level nets with a smaller number of places. Analysis at the subnet level is propagated to the higher level net. However, the firing/operation of the total net is not simplified by the identification of the node clusters which constitute the subnets. A PN may be modified by substituting a transition or place by the corresponding subnet or other way around. In addition, fusion of places and transitions can reduce the net.

Figure 3.8 shows a situation when a part of the Petri net is replaced by a single transition. Places P_2 and P_4 and transitions t_2 and t_3 are replaced by a single transition t_0.

3.3 Implementation

Major aspects of automation in a manufacturing plant include computer controlled machines and computer integrated manufacturing with the ability to efficiently schedule operations based on events, both expected and unexpected [6, 12, 30, 38]. In developing such an automated environment, one may often wish to try out scheduling solutions in a simulated environment [50] before moving to the actual production environment to remove as many unforeseen problems as possible before actual installation. Once the schedule is tested, it can be applied to the actual CIM system. A software environment for such a task [24, 59] is based on the model already shown in Figure 3.2 and consists of the following components:

System Interface (Monitor) : A graphical user interface providing all necessary functionality needed to display and manipulate various data.

Workflow Analyzer : The main part the software which analyzes the system and produces a corresponding schedule and/or simulation.

System Knowledge : A database containing both general information about machines/resources and specific information for the current task.

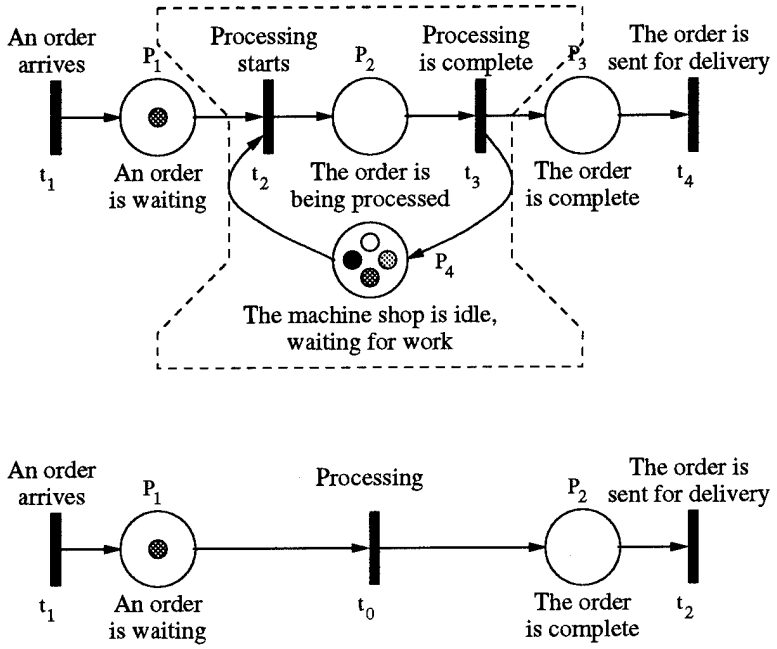


Figure 3.8: Substitution of transition

Figure 3.9 describes in more details the structure of each of the three main system parts.

3.3.1 System Interface

The system interface has been built using ViewKit^{TM} [35] (C++/Motif) in order to provide a well-defined and portable user interface. The plant description is given in a form of a graph where nodes represent machines and/or resources and edges represent connections between machines and/or resource utilization. A manufacturing plant is then defined by constructing a corresponding graph, from the library of known machines/resources stored in the object-oriented database (ObjectStore [11, 1]) as shown in Figures 3.10 and 3.11. A machine/tool is represented as a node in the graph, and flow of parts is described by arcs between nodes. Nodes are classified based on ASME standard 101 (operation, transportation, storage, delay, inspection), similar to JIS Z 8206.

The user interface is utilized as a system monitor. This is done as follows:

- Node appearance is related to the state of associated machine/tool
- Current schedule and flow of parts are represented by arcs in the graph
- Dialogs are used to represent various data and information about system
- Additional features (zooming, overview window, etc.)
- System operator interface

The System Interface includes the following:

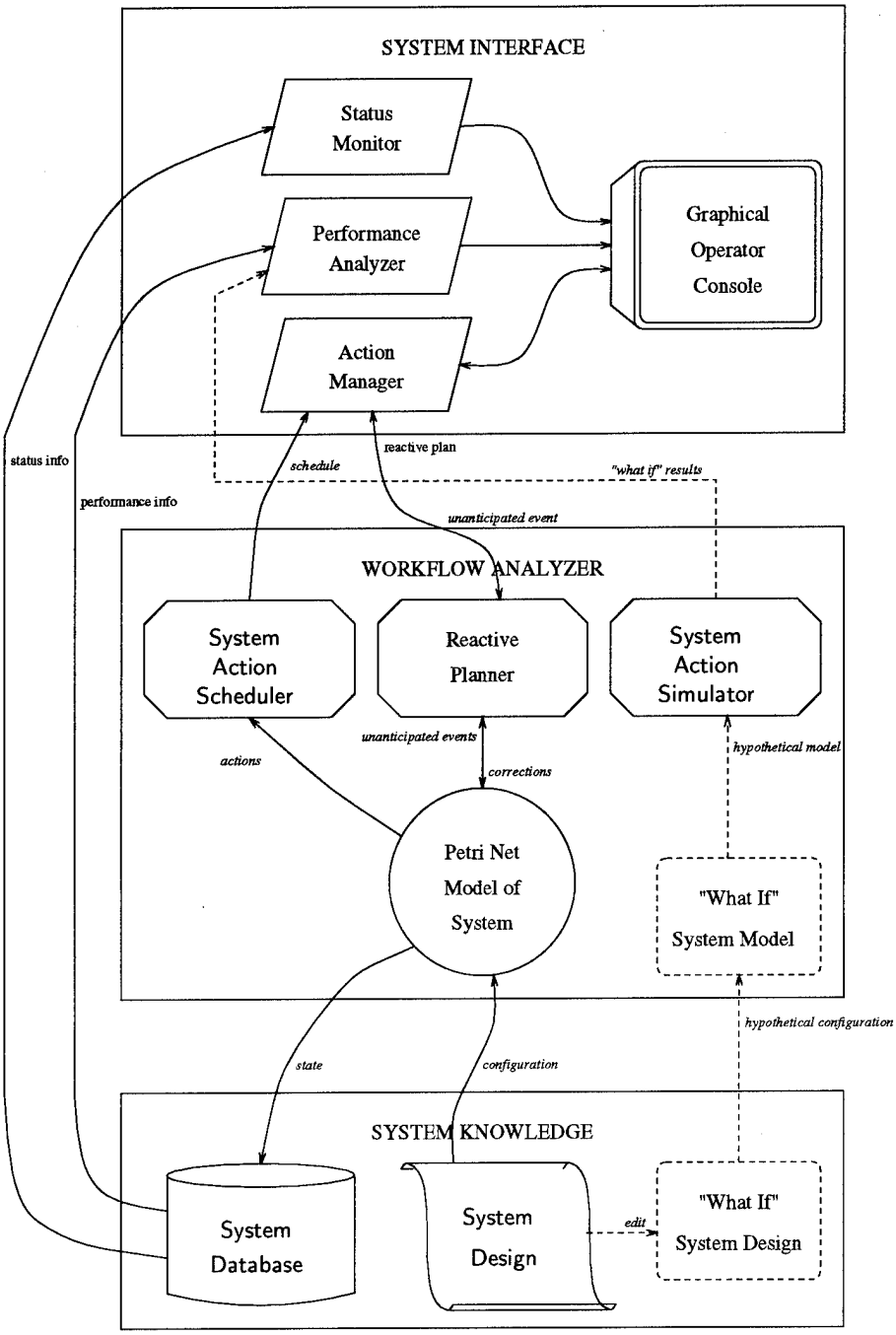


Figure 3.9: Overview of the system model

Graphical Operator Console : Actual display of the data and user interface.

Status Monitor : Based on the current state of the system Graphical Operator Console data display is updated.

Performance Analyzer : Reports (textual or graphical) are generated based on the current and previous states of the system. In addition, various "What if" scenarios can

be used to generate reports about alternative schedules/solutions.

Action Manager : New schedules and/or actions from the Workflow Analyzer are used to update the Graphical Operator Console. This module also serves as an interface to the CIM system thus providing on-line capabilities.

3.3.2 Workflow Analyzer

The Workflow Analyzer is the most important part of the DRAS scheduler. Its function is to evaluate the current state of the system and suggest possible actions in order to accommodate for changes, e.g., machine/operator errors or modifications in the production plan. It is implemented in C++ thus reflecting the object-oriented view of the components and data in the system.

A class library is used to represent characteristics of various objects and functions in the system. Well-defined taxonomy makes it easier to reuse existing data and to incorporate new data.

The Workflow Analyzer consists of:

Petri Net Model of the System : A high-level Petri net is used to model the system. Each object (machine, resource) is represented by its own Petri net and these nets are connected among themselves. The marking of the Petri net represent the state of the system.

System Action Scheduler : Based on the marking of the Petri net, a schedule is derived in order to get a sequence of actions which can be repeated over and over so that production can go on.

Reactive Planner : The Reactive Planner responds to errors coming form the Action Manager and updates the Petri net Model of System.

"What if" System Model : In order to provide "What if" functionality, the changes in the system design are modeled here.

System Action Simulator : The System Action Simulator performs various simulations based on the changes in the system design.

The core of this software system is the Petri net Scheduler. Theoretical methodologies and applications of PNs have been developed for modeling analysis and performance evaluations of production control systems, factory automation, and discrete event dynamic systems. Recently, PNs and their modifications have also been used as a tool for performance analysis and evaluation of decision-making organizations. Important properties that have contributed to the value of PNs as a modeling tool include model clarity and ease of representation, ability to model nondeterminism, conflicts, timing information, resource sharing, concurrency, parallelism, and control of asynchronous operations.

3.3.3 System Knowledge

The System Knowledge consists of three parts:

System Database : It contains the information about the current state of the system as well as the history, i.e. what activities took place in the system.

System Design : A "generic" knowledge of machine and resource types, scheduling and planing is stored here.

"What if" System Design : This part of the System Knowledge is used to create various "What if" situations, i.e. different states of the system which are then used for the simulation purpose.

The database implementation has undergone several revisions. The initial implementation used Oracle relational database [9, 14, 37] and an interface layer to the rest of the system. Later, the Oracle database was replaced by a local data structure which was finally replaced by the ObjectStore object-oriented database [1].

The objects in the database include:

Machine : Describes a type of machine with associated attributes. It also includes a list of supported operations, a list of errors, and a list of instances of this type o machine. Each instance has some attributes that makes it unique (serial number, position, etc.).

Operation : Describes a production step that is defined by its input parts, output parts, and production time. The production time is always adjusted by the Work Rate of the particular machine.

Error : Describes a possible malfunction that is defined by a set of parameters which include mean repair time, probability of error, etc. The probability is always adjusted by the Failure Rate of the particular machine.

Part : Describes a product (or a semi-product) used or produced by an operation.

Property : Describes an attribute (size, color, etc.) associated to a part.

Production : Describes a set of input parts, a set of operations and a final part.

Figure 3.10 shows the high-level machine subschema for the DRAS system. The ellipses in Figure 3.10 represent non-primitive classes from the DRAS class library. The links between the classes represent attributes and are labeled by the attribute name. A link from class X to class Y labeled P means that P is an attribute of X with range class (domain) Y. Only non-primitive classes and attributes are shown in the figure. Attributes such as "machine name" with primitive values (string, integer, etc.) are not shown. From the machine point of view, the database is concerned with information such as machine dimensions, power requirements, computer connections, connectivity constraints, and power requirements. These attributes are consulted when placing a machine in the virtual environment to make sure no constraints have been violated and to ensure feasible machine configurations. The figure also shows a partial inheritance hierarchy for machines. Sewing Machines, Cutters, and Spreaders are all

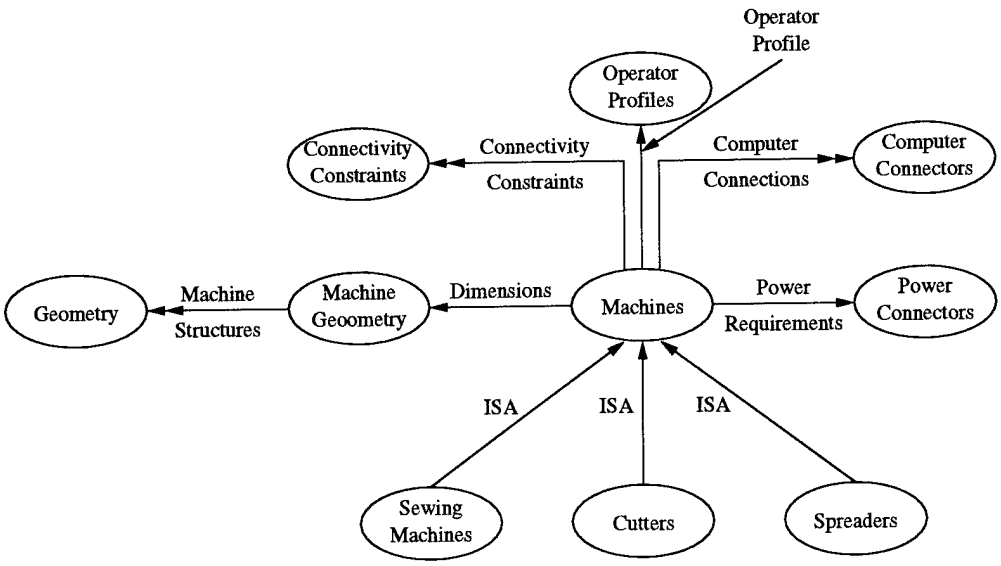


Figure 3.10: Machine Subschema

subclasses of Machines inheriting both structure and behavior. Sewing Machines, Cutters, and Spreaders define additional attributes and methods needed to more closely capture the state and behavior of these machines. This structure allows for easy extension of the schema by adding new types of machines as subclasses of the Machine class.

A high-level apparel plant schema is shown in Figure 3.11. As in Figure 3.10, only non-primitive classes are shown. With respect to an apparel plant, the database stores and retrieves data for the plant's electrical, lighting, and heating and cooling systems, computer networks, and plant dimensions and shape. There are also relationships between machines

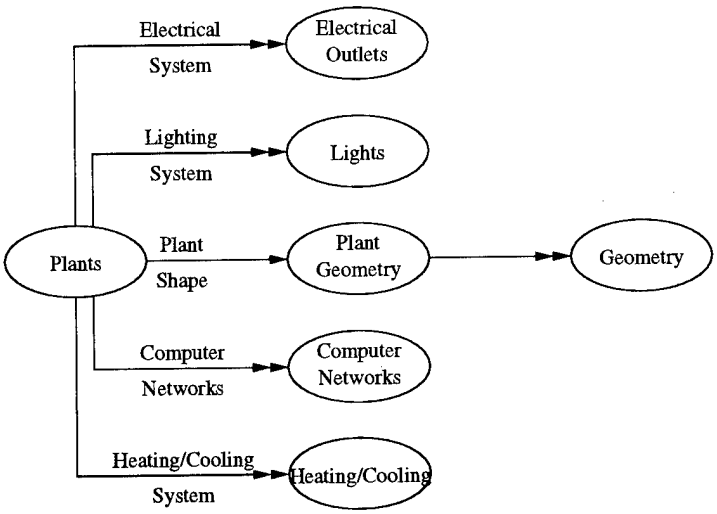


Figure 3.11: Plant Subschema

and plants. For example, Machine M is found in Plant P; Plant P contains Machine M. There

are many other relationships between plants and machines. This particular example shows the need for the relationship utility offered by the underlying object-oriented database. This relationship management helps to ensure that the database remains in a consistent state.

The Petri nets are used to model the system and object-oriented databases are used to store the knowledge about the system The combination of the Petri net model and the object-oriented database is used to implement the DRAS system. The basic structure shown in Figure 3.2 is implemented in three layers. The first layer is an interface layer that coordinates the communication with the user and the CIM system with the rest of the DRAS system. The second layer is a "Petri net" layer that models the CIM system and produces the schedule. The third layer is a database layer that stores all the necessary knowledge about the CIM system and the production process.

3.4 Example

A sample system is shown in Figure 3.12 where two different parts are used to assemble a final product. Before the assembly, both parts are stored (Storage A and B). They are then transported (Transportation A and B) to preprocessing (Operation A and B) and then transported (transportation C an D) to the assembly (Operation C). Once parts are assembled, the final product is transported (Transportation E) to the warehouse (Storage C).

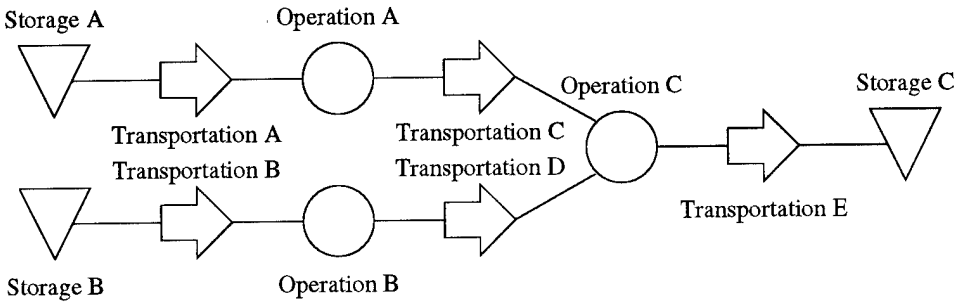


Figure 3.12: A sample system

All storage, transportation and operation nodes represent a machine/tool. Each machine is represented by the corresponding Petri net (Figure 3.14). These Petri nets are then joined together by transitions that correspond to the connecting lines. For each type of node in Figure 3.12 there is a corresponding colored Petri net (Figure 3.13). The creation of such a Petri nets is automated based on the information about the machine. The information required to construct the colored Petri nets is stored in the DRAS database.

The differences between Petri nets for different nodes (machines) is both in marking and structure. Petri nets for Operation A and Operation B have the same structure, however since the corresponding machines process different parts, markings are different because of the different colors for tokens. Petri nets for Operation A and Operation C have different structures because for Operation C there are two input transitions t_1 and places P_1 (Figure 3.4). Combining all Petri nets together by joining matching input and output transitions

a Petri net for the whole system is constructed.

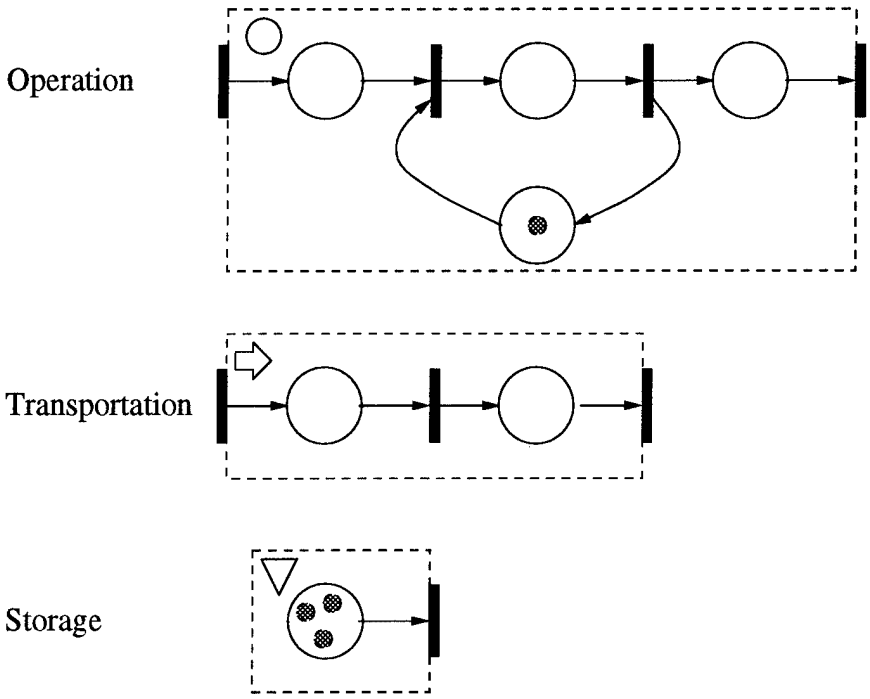


Figure 3.13: Petri nets for nodes

The complete Petri net is shown in Figure 3.12. This Petri net models the whole system and serves as a base for the schedule. The change in system is introduced as a change in marking. This change is reflected through change in the properties of the Petri net (e. g. liveness). If, for example, a machine for Operation B breaks down, that is modeled by removing a token either from place P_2 or P_4 (Figure 3.4), based on the current marking. The newly created marking is then analyzed and as a result, transitions t_2 and t_3 in colored Petri net for Operation C are no longer live. This means that current production has stopped. Then, a decision is made, based on the available information.

- If the machine has a short repair time, a possible action is to do nothing and wait for the repair to be completed which will be modeled by puting back the token that has been removed because of the machine breakdown.
- If there is an additional resource available (machine), the system may be reconfigured and production continues as before.
- If there is an alternative production available which includes only Operation B and C, the system may be reconfigured and the alternative production is started.

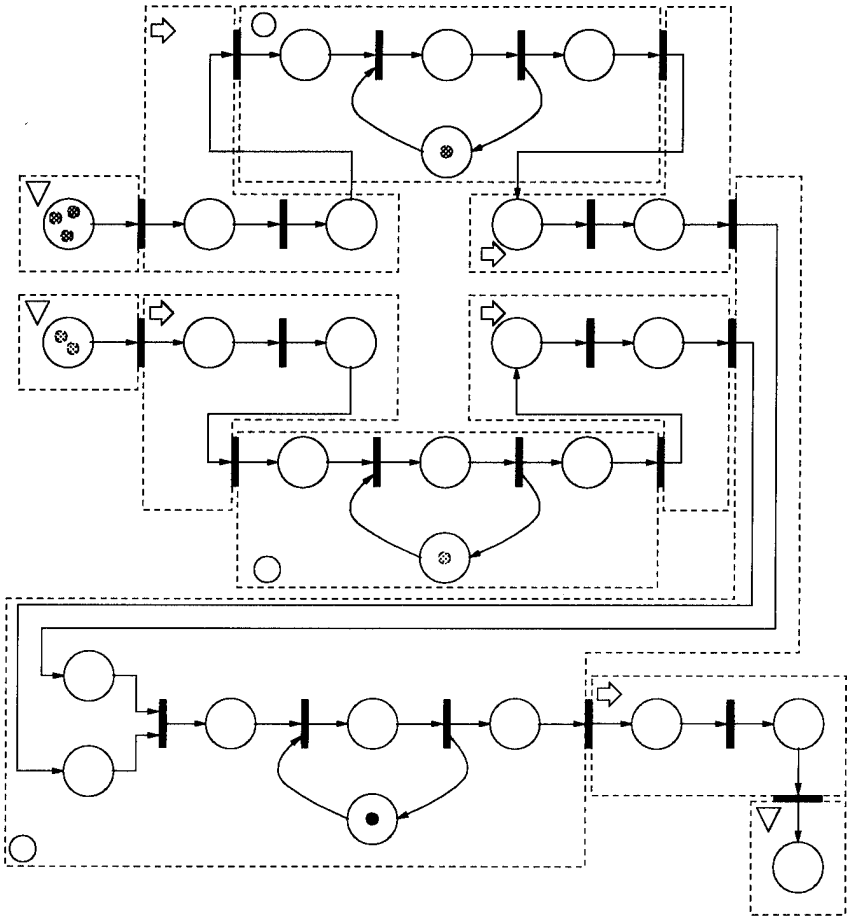


Figure 3.14: Petri net for the sample system

Chapter 4

User Guide

4.1 Introduction

The DRAS software is intended to be the user friendly and easy to use. Program is started by double clicking on icon representing either the DRAS application (dras) or a file conating DRAS data (example.dras), as shown in Figure 4.1.

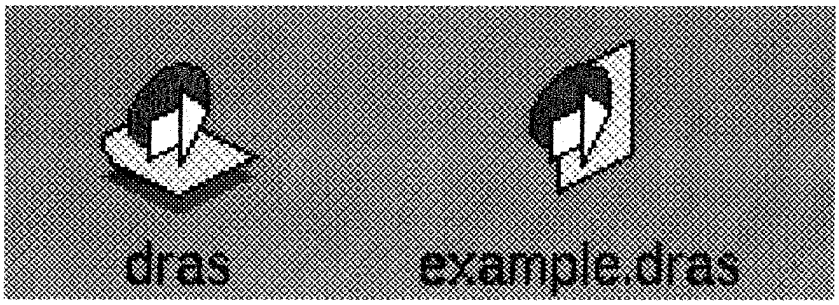


Figure 4.1: Icons

Once the application is running, it display the factory layout, as defined in the file. A menu bar (Figure 4.2) provides a set of menus that provide necessary functionality to use the DRAS system. Nodes' positions in the application window can be set independently of

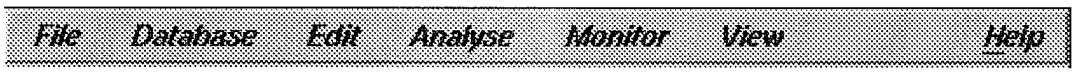


Figure 4.2: Menu bar

the actual position in the factory (Figure 4.3). Following sections describe in more details each menu and its items (file, database, edit, analyze, monitor, view, help).

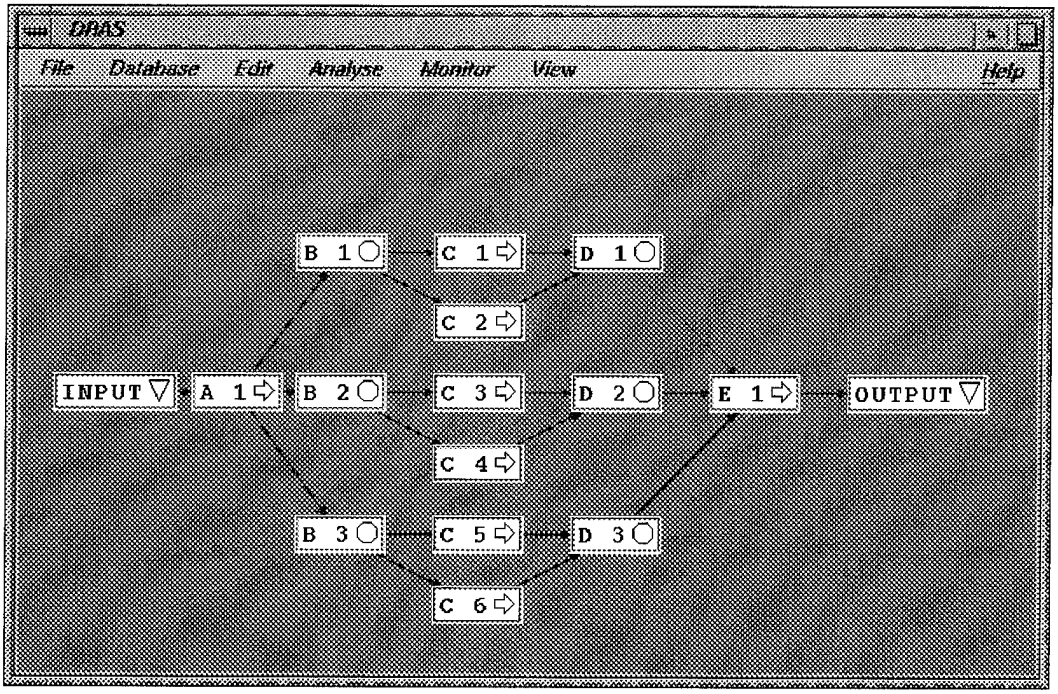


Figure 4.3: User interface

4.2 File Menu

The file menu (Figure 4.4) provides functionality related to the file manipulation.

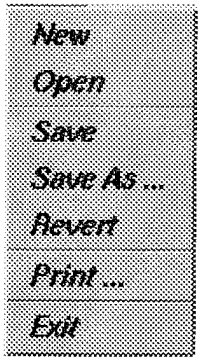


Figure 4.4: File menu

New : Deletes existing layout data and creates new, untitled layout. Before deleting, the user is asked to confirm the operation.

Open : Opens and existing file containing DRAS related data. Double clicking on the DRAS file icon is a shortcut for this operation. If there is already an open file which has been modified since the last save operation, the user is asked whether to save the old file.

Save : Save the current layout (if needed). If the current layout has no name (new file), the user is asked to enter the file name.

Save As : Save the current layout under a different name.

Revert : Undo all changes since the last save operation. Before undo, the user is asked to confirm the operation.

Print Prints the current layout.

Exit Exits the application. If there is a data that has noe been saved, user is asked whether to save it.

An example of the dialogs is given in Figure 4.5.

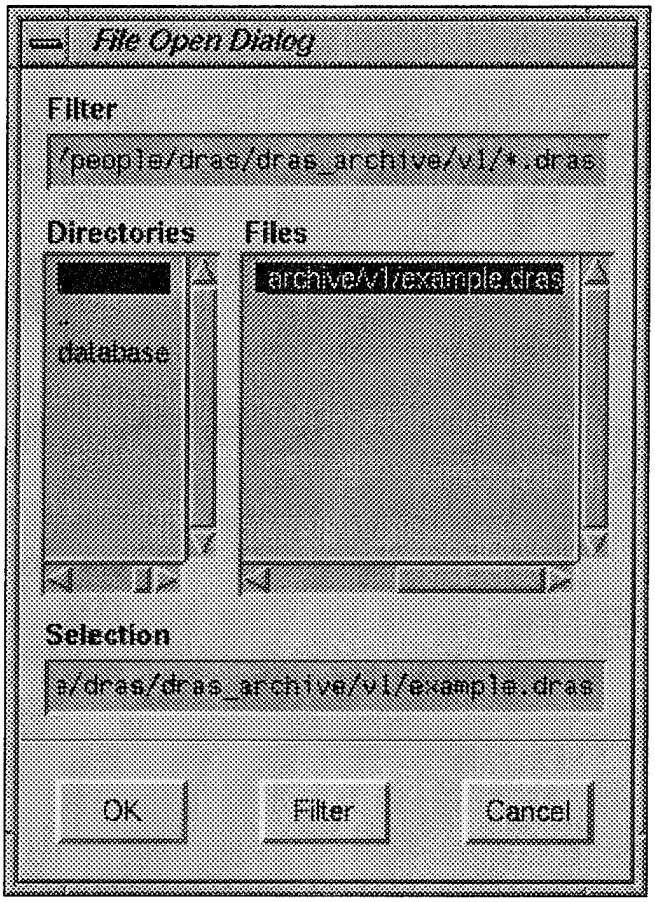


Figure 4.5: File open dialog

4.3 Database Menu

The database menu (Figure 4.6) provides access to the database. The database contains

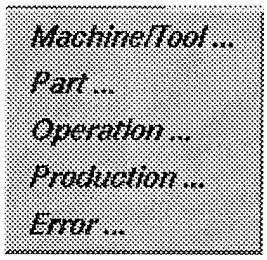


Figure 4.6: Database menu

all the necessary system knowledge, as well as current layout. Therefore, the user can view/modify information related to:

Machine/Tool : Node in the layout.

Part : Elements that are contained in a node.

Operation : Current state of the node.

Production : Overall production plan for that layout.

Error : Posiible types of errors.

When an item for the database menu is selected a dialog shown in Figure 4.7 appears. By selecting the appropriate category, the user can access other database components without going through the database menu again.

In addition, the user can access some layout node related data by double clicking on that node. This displays the node dialog shown in Figure 4.8. The dialog that appears also enables modifying properties of the node.

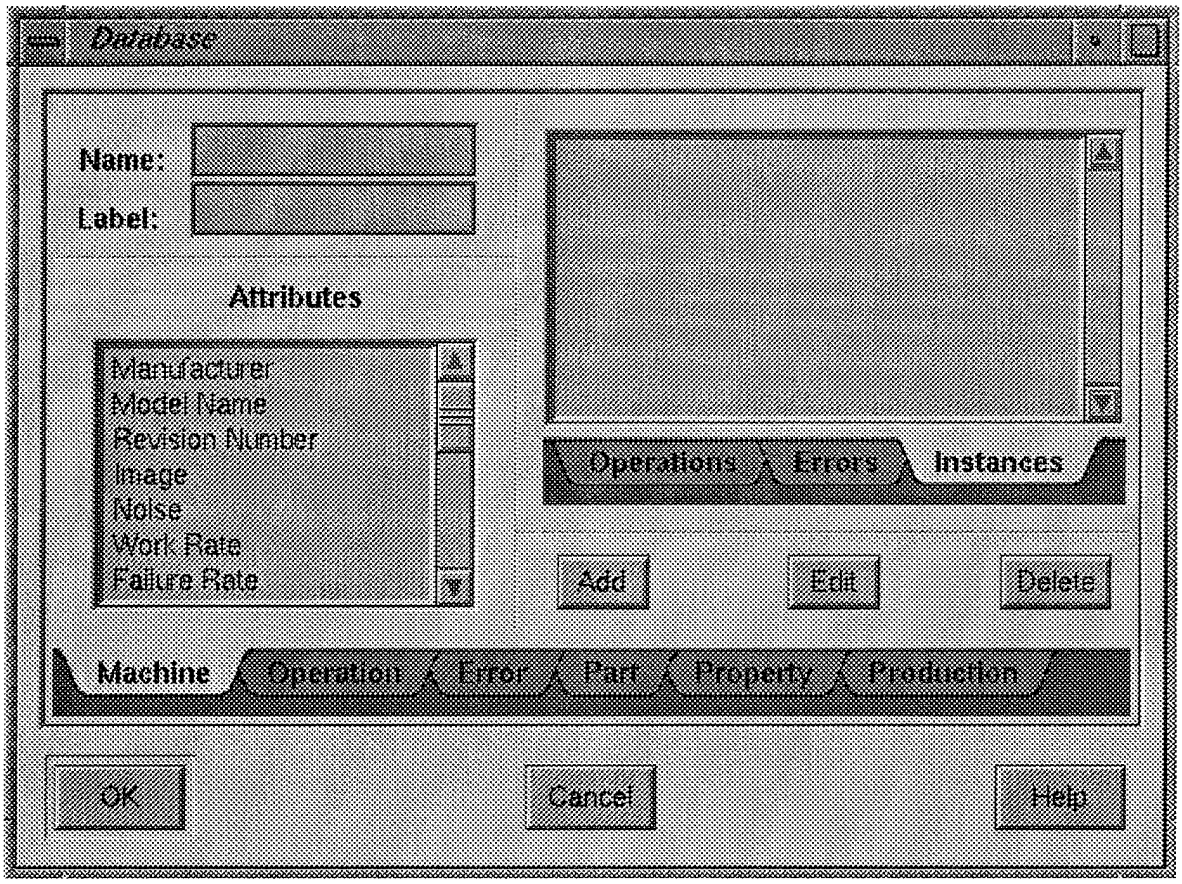


Figure 4.7: Database interface

4.4 Edit Menu

The edit menu (Figure 4.9) provides standard editing functions.

Undo : Undo the last cut or copy operation.

Cut : Remove the selected item.

Copy : Copy the selected item.

Paste : Paste the content of the last copy operation.

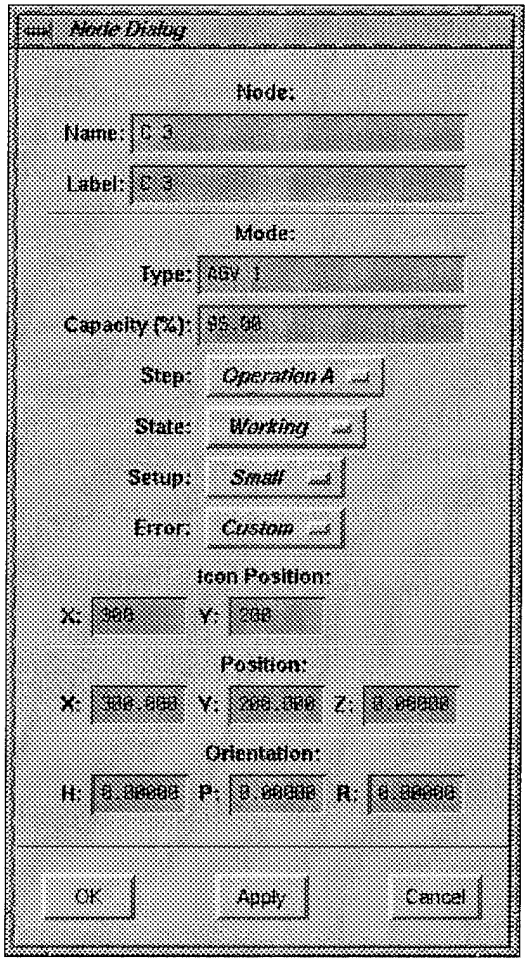


Figure 4.8: Node Dialog

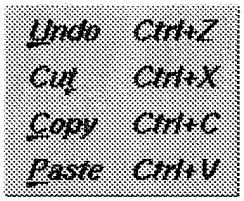


Figure 4.9: Edit menu

4.5 Analyze Menu

The analyze menu (Figure 4.10) provides for the checking of the system operation and rescheduling of the production plan.

Schedule : In the case of an error, an alternative schedule is generated and displayed. As an example, a machine error is indicated by red color of the node C 3 (Figure 4.11). By activating this menu item, a new schedule is generated (Figure 4.12).

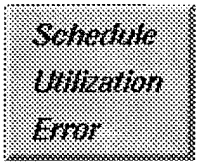


Figure 4.10: Analyze menu

Utilization : For selected resources (machine/tool) utilization is analyzed.

Error : Review and analysis of errors that occur in the system.

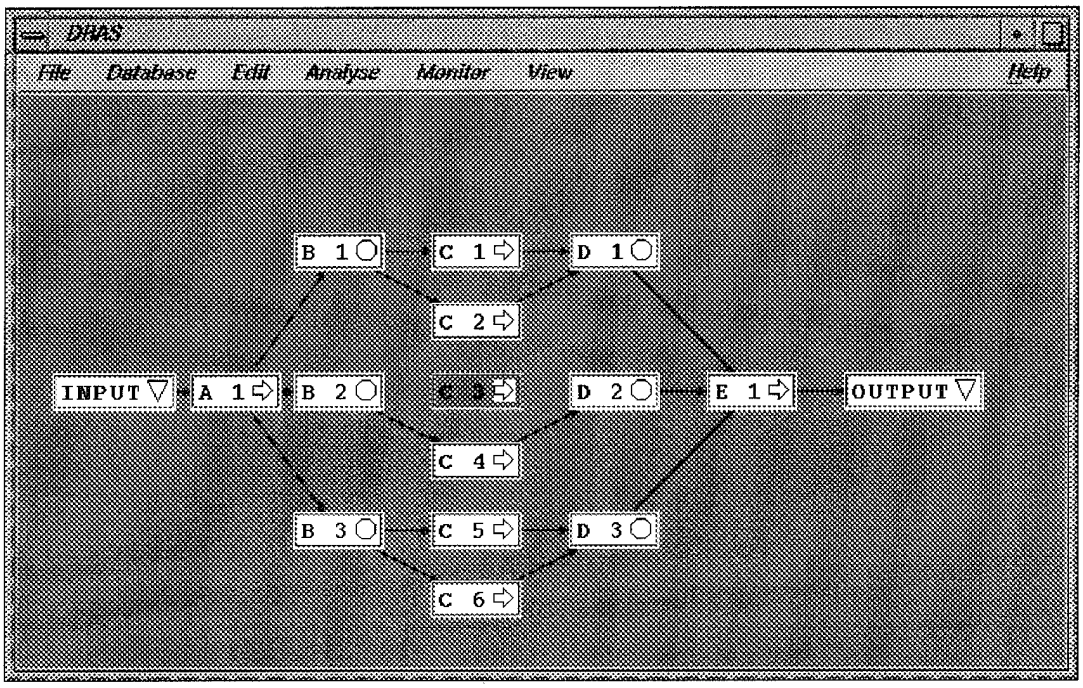


Figure 4.11: Node failed

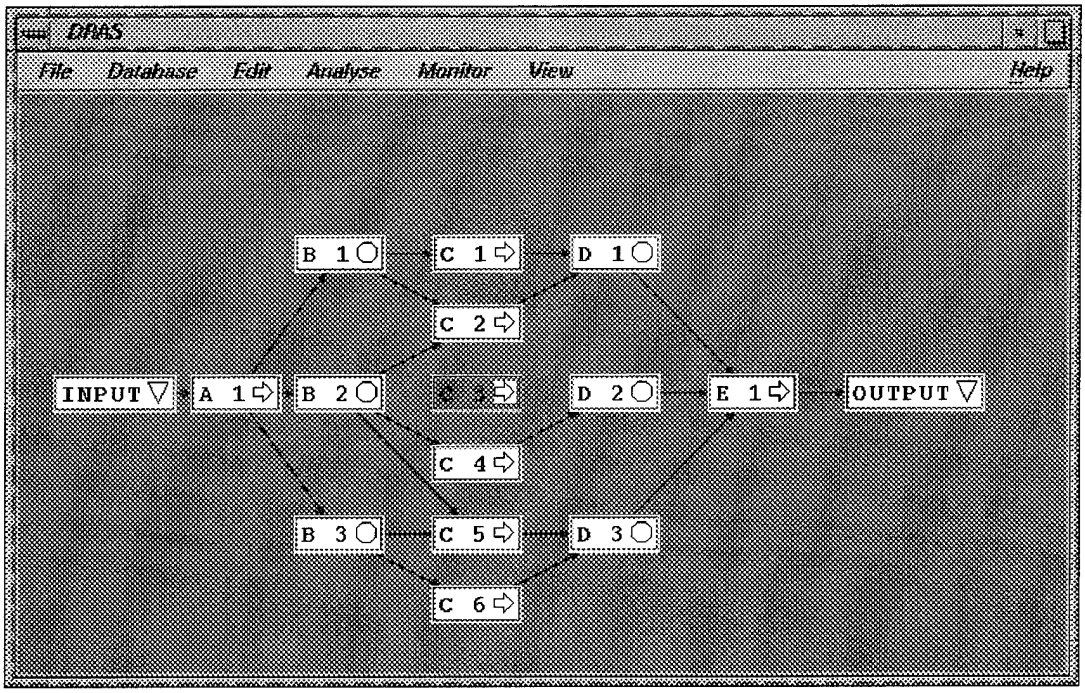


Figure 4.12: New Schedule

4.6 Monitor Menu

The monitor menu (Figure 4.13) enables monitoring functions that are not directly related to the scheduling.

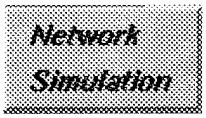


Figure 4.13: Monitor menu

Network : Controls the connection to the cell controller and the real-time payroll system. Due to the unavailability of this components, a small application has been written to emulate their behavior and to communicate messages as if they are running.

Simulation : "What-if" scenarios are tested by switching to the simulation mode where the "dry" run of the system is performed.

4.7 View Menu

The view menu (Figure 4.14) enables viewing various aspects of the system in different formats.

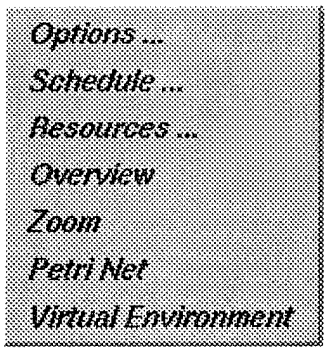


Figure 4.14: View menu

Options : Displays and edits various application options.

Schedule : Displays current schedule.

Resources : Displays resources usage.

Overview : For large layouts, a small window displaying a layout overview is shown (Figure 4.15).

Zoom : Scales up or down the layout displayed in the application window.

Petri Net : A Petri net of the system is displayed.

Virtual Environment : Display a virtual environment for the given layout (Figure 4.16). This has very limited functionality.

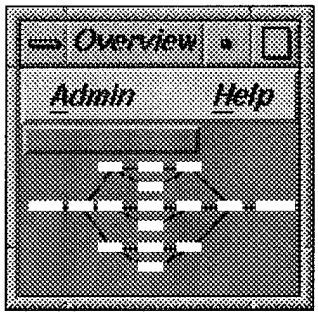


Figure 4.15: Overview window

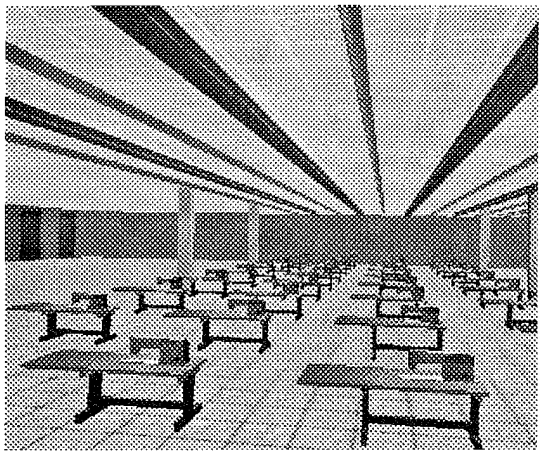


Figure 4.16: Virtual Environment

4.8 Help Menu

The help menu (Figure 4.17) provides some limited help to the user while using the program.

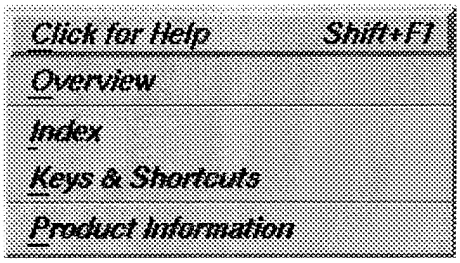


Figure 4.17: Help menu

Click for Help : Information about the part of the application window selected by next click is displayed.

Overview : General information about the application (text window).

Index : List of selected topics with additional information.

Keys & Shortcuts : Not functional, present only to comply with the standard interface.

Product Information : A product information dialog is shown (Figure 4.18).

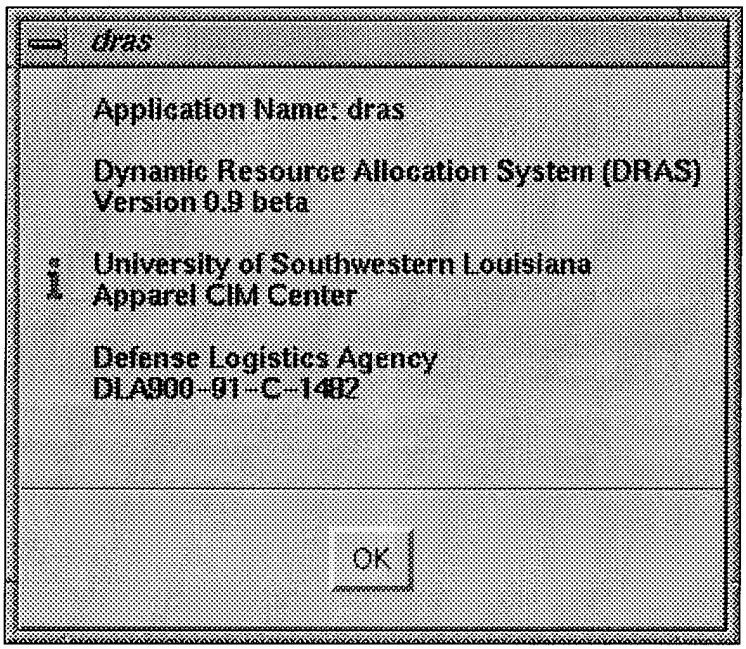


Figure 4.18: DRAS product information dialog

Chapter 5

Conclusion

The DRAS scheduler has addressed the need for flexibility in apparel manufacturing processes:

- by providing the means for flexible support of automated machines through a CIM network
- by providing the necessary communication to inform the machines that a particular bundle has arrived for processing as well as giving the details of the required processing,
- by facilitating a more flexible approach to manufacturing since the machine settings can be adjusted as frequently as necessary, and,
- by allowing the use of small bundles and supporting changes in style.

The combination of the CIM system network and the DRAS results in a CIM implementation that assists in quick response to demands and aids in rapid changeover by accommodating variation in product mix, improving operator performance, reducing management involvement in day-to-day production details and providing an enhanced information system to project Mean Time Between Failure (MTBF) and Mean Time to Recover (MTTR) statistics [25].

One important area of further development is the application of the Virtual Reality based interface/environment. Some initial work has been done [5, 26, 65] which validates the concept. Current work is focused on the full integration of Virtual Reality that will include the integration of a multimedia object-oriented database which will be used for the interactive design and modeling in the virtual environment.

Bibliography

[1] —. *ObjectStore Tutorial: Release 3.0 For UNIX Systems.* Object Design, Inc., Burlington, MA 01803-4194, Dec. 1994. Part Number 300-100-002 3C.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modeling with Generalized Stochastic Petri Nets.* Wiley Series in Parallel Computing. John Wiley & Sons, Inc., Chichester, 1995.

[3] R. Y. Al-Jarr and A. A. Desrochers. Petri nets in automation and manufacturing. Technical Report RAL99, Rensselaer Polytechnic Institute, Troy, NY, Nov. 1987.

[4] R. Y. Al-Jarr and A. A. Desrochers. Modeling and analysis of transfer lines and production networks using generalized stochastic Petri nets. In *Proceedings of the Conference on University Programs in Computer Aided Engineering, Design and Manufacturing,* Atlanta, Georgia, June 1988.

[5] L. Albright, D. Moreau, and T. Williams. A virtual reality framework for shopfloor configuration. In *Proceedings of the Fifth Annual Academic Apparel Research Conference,* pages 7–1:7–9, Lafayette, Louisiana, Feb. 1994.

[6] R. G. Askin and C. R. Standridge. *Modeling and Analysis of Manufacturing Systems.* John Wiley & Sons, Inc., New York, 1993.

[7] C. Beck and B. Krogh. Models for simulation and discrete control of manufacturing systems. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation,* pages 305–310, Mar. 1986.

[8] J. Billington. Many-sorted high level nets. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models,* Kyoto, Japan, Dec. 1989.

[9] S. M. Bobrowski. *Mastering Oracle7 & Client/Server Computing.* Sybex, San Francisco, 1994.

[10] G. Bruno and G. Marchetto. Process-translatable Petri nets for the rapid prototyping of process control systems. *IEEE Transactions on Software Engineering,* SE–12, Feb. 1986.

[11] R. G. G. Cattell, editor. *The Object Database Standard: ODMG-93 Release 1.1.* Morgan Kaufmann Publishers, Inc., San Francisco, 1994.

[12] R. B. Chase and N. J. Aquilano. *Production & Operations Management: A Life Cycle Approach.* IRWIN, Homewood, IL 60430, sixth edition, 1992.

[13] G. Ciardo, J. Muppala, and K. S. Trivedi. SPNP: Stochastic Petri net package. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, Kyoto, Japan, Dec. 1989.

[14] P. Corrigan and M. Gurry. *ORACLE Performance Tuing.* O'Reilly & Associates, Inc., Sebastopol, CA 95472, 1993.

[15] D. H. Crocket and A. A. Desrochers. Manufacturing workstation control using colored Petri nets. Technical Report RAL83, Rensselaer Polytechnic Institute, Troy, NY, Aug. 1986.

[16] R. David and H. Alla. *Petri Nets and Grafcet: Tools for modelling discrete event systems.* Prentice-Hall, New York, 1992.

[17] A. A. Desrochers, editor. *Modeling and Control of Automated Manufacturing Systems.* IEEE Computer Society Press, Washington, D.C., 1990.

[18] A. A. Desrochers and R. Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis.* IEEE Press, New York, 1995.

[19] D. Dubois and K. Stecke. Using Petri nets to represent production processes. In *Proceedings of the 22nd CDC*, pages 1062–1067, Dec. 1983.

[20] P. A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds.* Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1995.

[21] H. J. Genrich. Predicate transition nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, Berlin, 1987.

[22] H. J. Genrich and K. Lautenbach. System modeling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[23] J. Gentina and D. Corbeel. Colored adaptive structured Petri nets: A tool for the automatic synthesis of hierarchical control of FMSs. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pages 1166–1173, Apr. 1987.

[24] D. Gračanin and P. Srinivasan. Software environment for simulation and real-time coordination of manufacturing systems. In *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, volume 2, pages 1544–1547, Lafayette, Louisiana, Aug. 1994.

[25] D. Gračanin, K. P. Valavanis, S. A. Smith, Jr., T. Williams, II, and A. Steward. A dynamic resource allocation system for computer integrated manufacturing systems. In *Proceedings of the Sixth International Symposium on Robotics and Manufacturing*, 1996. in print.

[26] D. Gračanin and T. Williams. A virtual reality based interface to a dynamic resource allocation scheduler. In *Proceedings of the 1995 IEEE International Symposium on Intelligent Control*, pages 254–258, Monterey, California, Aug. 1995.

[27] I. Hatono, N. Katoh, K. Yamagata, and H. Tamura. Modeling of FMS under uncertainty using stochastic Petri nets. In *Proceedings of the International Workshop on Petri Nets and Performance Models*, Kyoto, Japan, Dec. 1989.

[28] H. P. Hillion. Performance evaluation of decisionmaking organizations using timed Petri nets. Technical Report LIDS–TH–1590, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, Aug. 1986.

[29] P. Huber, K. Jensen, and R. M. Shapiro. Hierarchies in coloured Petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1990*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1990.

[30] J. Ishiwata. *IE for the Shop Floor*, volume 1: Productivity Through Process Analysis. Productivity Press, Inc., Portland, Oregon, 1991.

[31] K. Jensen. Coloured Petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, Berlin, 1987.

[32] K. Jensen. Coloured Petri nets: A high level language for system design and analysis. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets: Theory and Applications*, pages 44–119. Springer-Verlag, Berlin, 1991.

[33] K. Jensen and G. Rozenberg, editors. *High-level Petri Nets: Theory and Applications.* Springer-Verlag, Berlin, 1991.

[34] A. Johnson and M. Malek. Rainbow nets for system analysis. Technical Report TR51.0565, University of Texas at Austin, 1989.

[35] K. Jones. *IRIS ViewKit^{TM} Programmer's Guide.* Silicon Graphics, Inc., Mountain View, CA 94039-7311, 1994. Document Number 007-2124-001.

[36] M. Kamath and N. Vishwanatham. Applications of Petri net based models in the modeling and analysis of FMSs. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Apr. 1987.

[37] G. Koch. *ORACLE7: The Complete Reference.* Osborne McGraw-Hill, Inc., Berkeley, 1993.

[38] L. J. Krajewski and L. P. Ritzman. *Operations Management.* Addison-Wesley Publishing Company, Reading, Massachusetts, third edition, 1993.

[39] B. Krogh and R. Sreenivas. Essentially decision free Petri nets for real time resource allocation. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pages 1005–1011, Apr. 1987.

[40] J. Martinez, P. Muro, and M. Silva. Modeling, validation and software implementation of production systems using higher level Petri nets. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pages 1180–1185, Apr. 1987.

[41] K. McDermott and K. Kamisetty. Development of an industrial engineering based flexible manufacturing system. *Industrial Engineering Magazine*, Dec. 1991.

[42] R. McHaney. *Computer Simulation: A Practical Perspective.* Academic Press, San Diego, 1991.

[43] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr. 1989.

[44] T. Murata and D. Zhang. A predicate-transition net model for parallel interpretation of logic programs. *IEEE Transactions on Software Engineering*, 14(4):481–497, Apr. 1988.

[45] Y. Narahari and N. Vishwanatham. On the invariants of colored Petri nets. volume 222 of *Lecture Notes in Computer Science*, pages 330–345. Springer-Verlag, Berlin, 1986.

[46] K. M. Passino and P. J. Antsaklis. Planning via heuristic search in a Petri net framework. In *Proceedings of the 1988 American Control Conference*, Atlanta, GA, June 1988.

[47] K. M. Passino and P. J. Antsaklis. Planning via heuristic search in a Petri net framework. In *Proceedings of the Third IEEE International Symposium on Intelligent Control*, Arlington, VA, Aug. 1988.

[48] J. L. Peterson. Petri nets. *ACM Comput. Surv.*, 9:223–252, Sept. 1977.

[49] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1981.

[50] U. W. Pooch. *Discrete Event Simulation: A Practical Approach.* CRC Press, Boca Raton, Florida, 1993.

[51] C. Ramchandani. Analysis of asynchronous concurent systems by timed Petri nets. Technical Report 120, Laboratory of Computer Science, MIT, Cambridge, MA, 1974.

[52] W. Reisig. *Petri Nets: An Introduction.* Springer-Verlag, Berlin, 1985.

[53] W. Reisig. *A Primer in Petri Net Design.* Springer Compass International. Springer-Verlag, Berlin, 1992.

[54] P. Remy, A. Levis, and V. Jin. On the design of distributed organizational structures. Technical Report LIDS–P–1581, LIDS, MIT, Cambridge, MA.

[55] A. Sahraoui, H. Atabakche, M. Courvoisier, and R. Valette. Joining Petri nets and knowledge based systems for monitoring purposes. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Apr. 1987.

[56] M. Schiffers and H. Wedde. *Analyzing Program Solutions of Coordinated Problems by CP Nets*, volume 64 of *Lecture Notes in Computer Science*, pages 463–473. Springer-Verlag, Berlin, 1987.

[57] M. Silva and R. Valette. *Petri Nets in Flexible Manufacturing*, pages 375–417. Advances in Petri Nets. Springer-Verlag, Berlin, 1990.

[58] S. A. Smith, Jr., T. Williams, II, and P. M. Landry. Transfer of CIM technology to an apparel manufacturing plant. In *Proceedings of the Fifth Annual Academic Apparel Research Conference*, pages 11.1–11.14, Lafayette, Louisiana, Feb. 1994.

[59] P. Srinivasan, D. Gračanin, and D. Brockhaus. Modeling complex information systems using parameterized Petri nets (PPNs). In *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, volume 1, pages 735–738, Lafayette, Louisiana, Aug. 1994.

[60] D. Tabak and A. Levis. Petri net representation of decision models. *IEEE Transactions on Systems, Man and Cybernetics*, SMC–15(6):812–818, 1985.

[61] K. P. Valavanis. On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended Petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 20(1):94–110, January/February 1990.

[62] F. Y. Wang and G. N. Saridis. A formal model for coordination of intelligent machines using Petri nets. In *Proceedings of the Third IEEE International Symposium on Intelligent Control*, Arlington, VA, Aug. 1988.

[63] F. Y. Wang and G. N. Saridis. The coordination of intelligent machines: A case study. In *Proceedings of the Third IEEE International Symposium on Intelligent Control*, Albany, NY, Sept. 1989.

[64] S. Weingaertner. A model of submarine emergency decision making and decision aiding. Masters thesis, LIDS, MIT, Cambridge, MA.

[65] T. Williams, D. Gracanin, K. Valavanis, and A. Steward. Real-time scheduling in an event driven virtual environment. In *Proceedings of the Conference on Advances in Modeling and Simulation*, pages 3–12, Huntsville, Alabama, Apr. 1994. U. S. Army Missile Command, Department of Defense.

[66] Y. Yaw, F. Law, and W. Ju. The algorithm of a synthesis technique for concurrent systems. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, Kyoto, Japan, Dec. 1989.

[67] A. Zenie. Colored stochastic Petri nets. In *Performance'84*, 1985.

[68] M. Zhou. *A Theory for the Synthesis and Augmentation of Petri Nets in Automation.* PhD thesis, ECSE, Rensselaer Polytechnic Institute, Troy, NY, 1990.

[69] M. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Kluwer Academic Publishers, Boston, 1993.